Refine Search

Search Results -

Term	Documents
(42 AND 5).USPT.	3
(L5 AND L42).USPT.	3

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L47

Refine Search

Recall Text | Clear | Interrupt

Search History

DATE: Wednesday, April 21, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=ADJ			
L47	l5 and L42	3	L47
L46	l5.ab. and L42	0	L46
L45	l31 and L42	0	L45
L44	l22 and L42	0	L44
L43	l24 and L42	0	L43
L42	l20 and L41	42	L42
L41	l16 near5 L40	85	L41
L40	most significant near1 bit$1	25982	L40
L39	compar$ and L38	1	L39
L38	l35 and allow$	1	L38
L37	l35 and L36	0	L37
L36	(deny$ or permit$) and 35	485483	L36
L35	5802590.pn.	1	L35

h e b b cg b e e ch

L34	l9 and l20 and l26 and L32	14	L34
L33	l27 and L32	0	L33
L32	l30 and L31	141	L32
L31	resource$1 near2 creat$	1404	L31
L30	time stamp$	10202	L30
L29	l21 and L28	0	L29
L28	l2 and L27	9	L28
L27	l20.ab. and L26	91	L27
L26	l9 and L25	391	L26
L25	l20 and L24	762	L25
L24	resource$ near requested	973	L24
L23	l21 and L22	5	L23
L22	shared resource$1 near3 requested	51	L22
L21	number near1 time$1	104979	L21
L20	access$ near5 (resource$ or memor$)	152311	L20
L19	l17 and L18	1	L19
L18	resource$1	86086	L18
L17	l15 and L16	1	L17
L16	identifier$1	47857	L16
L15	l10 and l12	1	L15
L14	l2 and l10 and l12	0	L14
L13	l2 and l10 and L12	0	L13
L12	5421012.pn.	1	L12
L11	l3 and L10	0	L11
L10	version$1	192487	L10
L9	identifier and resource$1	17497	L9
L8	l4 and L7	5	L8
L7	l5 and L6	827	L7
L6	L5.ab.	827	L6
L5	access$ near3 resource$1	8648	L5
L4	version number and counter and (access$ near2 resource$1)	129	L4
L3	l1 and L2	1	L3
L2	counter$1	377547	L2
L1	5469556.pn.	1	L1

END OF SEARCH HISTORY

h e b b cg b e e ch

L18: Entry 3 of 11 File: USPT May 20, 2003

DOCUMENT-IDENTIFIER: US 6567919 B1
TITLE: Authenticated communication procedure for network computers

Abstract Text (1):
Authentication of a request by a computer for access to a resource is accomplished by means of a randomly generated password that can only be used a limited number of times. In a disclosed embodiment of the invention, a network computer sends a boot request to a network server. In response, the network server generates a random password, and sets a use counter to a value which indicates the number of times that the password can be used for access to network resources. This password is transmitted to the network computer, which uses it to initiate a session with a network file server, and access network resources. The network server then invalidates the password, by decrementing the use counter to zero. As a result, even if the password becomes known to an unauthorized user as it is being transmitted from the network server to the network computer, it cannot be improperly employed to gain access to any network resources.

Brief Summary Text (5):
As the sophistication and use of local area networks continues to evolve, more and more functions are being provided by centralized network servers, rather than the individual workstations themselves. For example, to facilitate backup and retrieval, it has become a common practice to store all data files on a network file server, rather than on local media at the individual workstations. Other types of services are also being handled at the network server, for example external connections to remote sites. Consequently, there has been a move toward so-called network computing, which reduces the need for fully-featured computers at each user location. Typically, the network computer at each user node in such a system might consist solely of a microprocessor, random access memory, input devices such as a keyboard and mouse, and a display monitor. All other devices associated with the computing environment, e.g., file storage, modems, printers, etc. are associated with, and controlled by, the network server. Since the individual terminals located at the user nodes contain a minimal amount of hardware, the cost of this type of arrangement is significantly less than a local area network which employs personal computers, or similar types of fully-featured computers, at the user nodes.

Brief Summary Text (9):
In accordance with the present invention, authentication of a request by a computer for access to system resources is accomplished by means of a randomly generated password that can only be used a limited number of times. In an exemplary embodiment of the invention, when a network computer is turned on, it sends a boot request to a network server. In response, the network server generates a random password, and stores the password in a file to indicate that it is associated with the network computer that issued the boot request. In addition, a use counter is set to a value which indicates the number of times that the password can be used for access to network resources, preferably once. This password is transmitted to the network computer, which then uses it to initiate a session with a network file server, and mount a storage volume. Once the storage volume has been mounted, the network computer has access to the files on that volume, and the password is no longer needed. Consequently, the network server invalidates the password, by setting the use counter to zero. As a result, even if the password becomes known to

an unauthorized user as it is being transmitted from the network server to the network computer, it cannot be subsequently employed to gain access to any network resources.

Detailed Description Text (14):
After confirmation that the requesting network computer has been previously registered, or after newly registering the network computer, the user's password is set to a randomly generated value, at step 44. In addition, a use count value is stored in connection with the password. The use count is set to a minimal number, to limit the number of times the newly generated password can be used to access network resources. FIG. 4 illustrates an example of various entries that might appear in the user registry 38. For each user, the registry contains a listing of its hardware address 46, user identification 48, assigned storage volume 50, password 52 and use count 54. In a preferred embodiment of the invention, when the password is generated, its use count is set to a value of one. In the particular example illustrated in FIG. 4, the first and third network computers in the registry, NC1 and NC3, are currently undergoing the boot process, and therefore their use count values are equal to one. All the other registered network computers have a use count value of zero, for reasons explained in detail hereinafter.

L18: Entry 7 of 11 File: USPT Aug 23, 1994

DOCUMENT-IDENTIFIER: US 5341491 A
TITLE: Apparatus and method for ensuring that lock requests are serviced in a multiprocessor system

Abstract Text (1):
A lockout avoidance circuit is provided for a plurality of nodes which generate lock requests for a shared resource such as a memory location. The circuit insures that lock requests are eventually satisfied. A lock queue includes a plurality of registers pipelined together. Lock requests only enter the lock queue if they are refused access to a shared resource a predetermined number of times. A first register is the head of the queue and the last register is the bottom of the queue. An enabling circuit allows the queue to store in the registers lock requests received from the different nodes in the order in which they are initially refused service. The enabling circuit operates the queue by pushing the stored lock requests toward the head of the queue each time the head entry in the queue is serviced. The lockout avoidance circuit is implemented at each level of the system wherein a lockout condition can occur.

Brief Summary Text (8):
The present invention overcomes these problems by providing an efficient and simple hierarchical system of lock out avoidance mechanisms for use in systems having central locations for receiving the lock requests. To avoid lock out each of the mechanisms uses a centralized queueing structure called a lock queue. The centralized queueing mechanism is used because all interlock signals, e.g. lock requests and unlock requests, are received at the central location. The lock queue contains as many entries as there are possible nodes from which lock requests can originate, but no one entry is specifically tied to any node. Each entry in the lock queue contains a valid indication field and a node identification (I.D.) field. The valid indication field indicates whether the lock request is valid and the node I.D. field stores the identification of the node generating the lock request. The lock queue further has a head and a tail entry. The head points to the first entry, i.e., the highest priority entry, in the queue and the tail points to the first available entry in the lock queue. Each time the head of the queue is serviced, i.e., its lock request is satisfied, the head of the queue moves to the next entry in line. If one of the nodes already has an entry in the queue which has not yet been serviced, then subsequent lock requests from that node will be written into the same place in the lock queue. This is possible because the entries only contain the node ID and not the request itself. In this manner, the lock queue need only contain a number of entries corresponding to the number of nodes from which lock requests can originate.

Brief Summary Text (9):
The lock queue is usually empty and lock requests, from the various nodes passing through the central location, only enter the lock queue, if, after a predetermined number of tries, they are unable to gain access to a particular resource. For example, assume that a lock request from node 0 is refused access because the particular memory location was already locked or a register for allowing the lock request was not available, then the central location increments a refusal counter for the particular node. When the node, in this case node 0, retries its lock request, it will then again be either refused or granted access. If the lock

request is refused a predetermined consecutive number of times while the lock queue is empty, then the node's I.D. will be stored at the head of the lock queue along with an indication of its validity.

Brief Summary Text (11):
In this manner, a lock out avoidance mechanism is operated such that if the lock queue is empty, it is difficult to enter the lock queue because a particular lock request must be refused the predetermined number of times before the lock request may enter the queue. Once in the queue, the lock requests are serviced serially which slows down the system. Therefore, the lock queue is operated to try to keep the queue empty by using the refusal counter. This operation ensures that the queue is usually empty. However, once the lock queue is no longer empty, all future lock requests are serviced strictly by their queue position. This operation continues until the lock queue again becomes empty. Thus, the lock out avoidance mechanism at this level guarantees that each node will eventually have its lock request serviced.

Detailed Description Text (23):
The write control logic 86 provides a LOCK REGISTER WRITE CONTROL signal to each of the registers 88 in the lock registers 80a-80n. Further, the write control block 86 provides a LOCK QUEUE REGISTER WRITE CONTROL signal to enable the multiplexers 96a-96d associated with state devices 94a-94d to operate in accordance with the lock queue 130 design. Also, a REFUSAL COUNTER INCREMENT CONTROL signal is provided from write control 86 to a plurality of refusal counters 131-136. Each refusal counter is associated with a node, such as CPU modules 0-3 (FIG. 1) in the system. The refusal counters 131-136 are incremented by signals from the write control 86. Once a particular refusal counter reaches a predetermined limit, an END OF COUNT signal is provided to the write control 86. The predetermined limit in the refusal counter is optimized for the best performance for lock requests in the system. The write control 86 further receives from the arbiter 51 (FIG. 2) a LOCK CLEAR and LOCK SET signal.

Detailed Description Text (28):
Referring back to FIG. 4, when a match is generated from one of the comparators 90 in one of the lock registers 80a-80n, a signal is sent over line 200 to the write control 86. The match indicates that the memory location is currently locked. The write control 86 then provides a REFUSAL COUNTER INCREMENT CONTROL signal. The REFUSAL COUNTER INCREMENT CONTROL signal enables the particular refusal counter 130-136 associated with the node generating the READ LOCK command which was refused. This signal increments the counter.

Detailed Description Text (29):
Similarly, when all of the lock registers 80a-80n are filled with a valid address and not the address for which the lock is requested, then the LOCK MISS signal provided from OR gate 82 to the write control 86 will enable the REFUSAL COUNTER INCREMENT CONTROL signal. This is possible because the write control 86 internally tracks the number of valid lock registers and therefore knows when the lock registers 80a-80n are filled.

Detailed Description Text (30):
Once a refusal counter 131-136 has been incremented a predetermined number of times, the END OF COUNT signal is provided to the write control 86. This generates a LOCK QUEUE REGISTER WRITE CONTROL signal from the write control 86 to the lock queue 130. The node source ID and the lock queue register valid bit (FIG. 4b) are then loaded and set, respectively, into the top of the queue 94d by enabling the multiplexer 96d for the particular READ LOCK command and associated address from bus 36. This lock request then becomes the head of the lock queue 130. Once the lock queue 130 contains a valid entry, any subsequent lock requests from other nodes will be automatically refused (regardless of whether the other conditions for refusing locks are satisfied) and will be stored in the queue 130.

Detailed Description Text (36):
An example of the lockout avoidance mechanism operation will now be given. Lock requests are generated from the various nodes (FIG. 1) in the form of READ LOCK commands and an associated address location and are received by the central unit 15 through their respective port logics 49a. The request passes through the scheduling logic 66 and into the lock logic 71 (FIG. 2) on bus 36. As long as the lock request is not refused by the lock logic 71, the request does not enter the lock queue 130 (FIG. 4). However, assuming the lock request from node 0, i.e., CPU 0, is refused because the particular location in memory has already been locked or because no free lock registers 80a-80n are available for allocation to the lock request, then the write control 86 increments the refusal counter 131-136 for the particular node, in this case CPU 0. After a predetermined period of time, CPU 0 again retries its lock request. If the lock request from CPU 0 is again refused, its respective refusal counter is incremented once more. Once a lock request has been refused a predetermined consecutive number of times while the lock queue 130 remains empty, the write control 86 then places the identification for the particular node at the head of the lock queue 130. Only the node ID (FIG. 4b) is stored in the lock queue and not the address of the lock request that was refused.

Detailed Description Text (38):
This operation assures that whenever the lock queue 130 is empty, it is difficult to enter the lock queue 130 because the lock requests from a particular node must be refused a predetermined consecutive number of times. Therefore, the lock queue 130 is predominantly kept empty, assuming the refusal counter is set at a high enough value. However, once the lock queue 130 has an entry, all future lock requests are serviced strictly by their lock queue position, i.e., the head of the queue gets satisfied first, the second entry is satisfied next, etc. This operation continues until the lock queue 130 is again empty.

Detailed Description Text (47):
The protocol used in the CF chip to avoid lockouts is essentially similar to the lock queue 130 used in the central unit. However, in this instance there is no need to implement a refusal counter in the CF chip 100, because the frequency at which lock requests are generated from the X chip at the CPU level is not great enough to require the use of a refusal counter.

CLAIMS:

2. A lockout avoidance circuit according to claim 1, wherein:

the means for counting refused lock requests comprises a plurality of counters coupled to said means for enabling, each one of said counters being associated with one of said nodes;

a count signal is provided to said means for enabling from one of said counters in response to the counter having reached a predetermined number of refused lock requests; and

said means for enabling stores a particular lock request from one of said nodes in the queue in response to the count signal associated with said node.

3. A lockout avoidance circuit according to claim 2 further comprising means for storing all subsequent lock requests from any of said nodes in the queue after the means for enabling has received the count signal from a counter.

6. A method for avoiding lockout of lock requests generated from a plurality of nodes coupled to a central location, the method comprising the steps of:

incrementing a counter associated with a particular one of the nodes when a lock

request from said particular node is refused in the central location;

generating an end of count signal from said counter when the lock request is refused a predetermined number of times;

storing the lock request as a head entry in a queue once the end of count signal is generated;

subsequently storing any further lock requests from different nodes as further entries in said queue once the queue contains a head entry; and

operating said queue to the entries in the order in which they are placed in the queue.

13. A circuit for servicing lock requests for memory locations generated from a plurality of nodes, comprising:

a plurality of lock registers, each of the lock registers including a storage device and a comparator having first and second inputs, said lock requests being provided to the first input of the comparator and to an input of the storage device, an output of the storage device being coupled to the second input of the comparator;

a lock queue coupled to said lock registers, the lock queue receiving lock requests at its inputs, the lock queue including a plurality of queue registers;

a control circuit coupled to the lock registers and lock queue for controlling the circuit operation;

a plurality of counters each providing a count signal to the control circuit, the number of counters and queue registers equalling the number of nodes from which lock requests are generated;

said control circuit including

a first signal for operating the lock registers;

a second signal for operating the lock queue; and

a third signal for operating the counters.

16. A circuit according to claim 15 wherein the lock refused signal enables the control circuit to generate the third signal to the respective counter associated with the node generating the lock request.

17. A circuit according to claim 16 wherein the counters are set to a predetermined value, said third signal incrementing the respective counter.

18. A circuit according to claim 17 comprising:

an end of count signal output from said counters once they reach their predetermined value, the end of count signal provided to the control circuit to activate the second signal.

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 27 of 27 returned.

☐ 1. Document ID: US 6636721 B2

L11: Entry 1 of 27 File: USPT Oct 21, 2003

US-PAT-NO: 6636721
DOCUMENT-IDENTIFIER: US 6636721 B2

TITLE: Network engineering/systems system for mobile satellite communication system

DATE-ISSUED: October 21, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Threadgill; Michael E.	Reston	VA		
Lin; ShihChao	Reston	VA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Mobile Satellite Ventures, LP	Reston	VA			02

APPL-NO: 09/ 918550 [PALM]
DATE FILED: August 1, 2001

PARENT-CASE:
RELATED APPLICATION This application is a continuation of U.S. application Ser. No. 08/931,622, filed Sep. 16, 1997 now U.S. Pat. No. 6,272,341 which in turn is a continuation of, and claims priority to, U.S. application Ser. No. 08/601,749, filed Feb. 15, 1996, now U.S. Pat. No. 5,713,075, which inturn claims priority from U.S. provisional application Serial No 60/007,804, entitled "Network Engineering and System Engineering System" filed Nov. 30, 1995, the details of which are incorporated herein by reference.

INT-CL: [07] H04 B 7/185, H04 Q 7/20

US-CL-ISSUED: 455/12.1; 455/427, 455/428
US-CL-CURRENT: 455/12.1; 455/427, 455/428

FIELD-OF-SEARCH: 455/427, 455/12.1, 455/428, 455/429

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

h e b b g e e e f e e ef b e

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5526404	June 1996	Wiedeman et al.	
5555444	September 1996	Diekelman et al.	
5586165	December 1996	Wiedeman	
5590395	December 1996	Diekelman	
5594740	January 1997	LaDue	
5594780	January 1997	Wiedeman et al.	
5713075	January 1998	Threadgill et al.	455/427
6185409	February 2001	Threadgill et al.	455/12.1
6272341	August 2001	Threadgill et al.	455/428
2002/0013149	January 2002	Threadgill et al.	455/427

OTHER PUBLICATIONS

"North America Mobile Satellite System Signaling Architecture", Lawrence White et al., American Institute of Aeronautics, Inc., 1992, pp. 427-439.
"The AMSC/TMI Mobile Satellite Services (MSS) System Ground Segment Architecture", J. Lunsford, et al., American Institute of Aeronautics and Astronautics, Inc., 1992, pp. 405-426.
"Call Control in the AMSC Mobile Satellite Services System", William R.H. Tisdale, et al., Pre-Publication Review Copy, American Institute of Aeronautics and Astronautics, Mar. 1, 1994, pp. 1-13.
"Westinghouse MSAT Mobile Terminal Channel Emulator", A. Fasulo et al., American Institute of Aeronautics and Astronautics, Inc., 1993, pp. 256-260.
"MSAT Network Communications Controller and Network Operations Center", Tony Harvey et al., American Institute of Aeronautics and Astronautics, Inc., 1993, pp. 270-260.
"MSAT and Cellular Hybrid Networking", Patrick W. Baranowsky II, Westinghouse Electric Corporation, 1993.
"Feederlink Earth Sation to Provide Mobile Satellite Service in North America", Robert H. McCauley, et al., American Institute of Aeronautics and Astronautics, Jan./Feb., 1994, pp. 1-9.
"Radio Transmission in the American Mobile Satellite System", Charles Kittiver, American Institute of Aeronautics and Astronautics, Inc., 1994, pp. 280-294.
"Summary of the AMSC Mobile Telephone System", Gary A. Johanson, et al., American Institute of Aeronautics and Astronautics, Inc., 1994, pp. 1-11.
"Implementation of a System to Provide Mobile Satellite Services in North America", Gary A. Johanson, et al., presented at International Mobile Satellite Conference '93, Jun. 16-18, 1993.
"The American Mobile Satellite Corporation Space Segment", David J. Whalen, et al., 1992, 99. 394-404.

ART-UNIT: 2683

PRIMARY-EXAMINER: Cumming; William

ABSTRACT:

A mobile satellite system includes a network engineering/systems engineering (NE/SE) system. The NE/SE performs the processes of comparing expected traffic loads with capability and availability of space and ground resources in the mobile satellite system, formulating tactical plans to maximize available resources of the satellite, and producing frequency plans for different geographical regions and defining circuit pools for different groups of users.

12 Claims, 45 Drawing figures

h e b b g e e e f e e ef b e

☐ 2. Document ID: US 6625447 B1

L11: Entry 2 of 27 File: USPT Sep 23, 2003

US-PAT-NO: 6625447
DOCUMENT-IDENTIFIER: US 6625447 B1

TITLE: Method and architecture for an interactive two-way data communication network

DATE-ISSUED: September 23, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Rossmann; Alain	Menlo Park	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Openwave Systems Inc.	Redwood City	CA			02

APPL-NO: 09/ 200359 [PALM]
DATE FILED: November 24, 1998

PARENT-CASE:
The present application is a continuation application of U.S. patent application Ser. No. 08/978,701, filed on Nov. 26, 1997, and entitled "A Method and Architecture for an Interactive Two-Way Data Communication Network," which is a continuation application of U.S. patent application Ser. No. 08/570,210, filed on Dec. 11, 1995, and entitled "A Method and Architecture for an Interactive Two-Way Data Communication Network," now issued as U.S. Pat. No. 5,809,415.

INT-CL: [07] H04 M 11/00

US-CL-ISSUED: 455/422; 455/412, 455/426, 455/517, 340/825.27, 709/217
US-CL-CURRENT: 340/825.27, 455/517, 709/217

FIELD-OF-SEARCH: 455/422, 455/426, 455/461, 455/414, 455/552, 455/566, 455/466, 455/412, 455/517, 709/217, 340/825.26, 340/825.27

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4787028	November 1988	Finfrock et al.	
4812843	March 1989	Champion, III et al.	
5008925	April 1991	Pireh	
5128672	July 1992	Kaehler	

h e b b g e e e f e e ef b e

5335276	August 1994	Thompson et al.	
5428823	June 1995	Ness-Cohn et al.	455/31.1
5465401	November 1995	Thompson	
5491745	February 1996	Roeder	
5543789	August 1996	Behr et al.	340/995
5548636	August 1996	Bannister et al.	
5555446	September 1996	Jasinski	455/31.1
5560008	September 1996	Johnson et al.	395/187.01
5577100	November 1996	McGregor et al.	
5577103	November 1996	Foti	
5577209	November 1996	Boyle et al.	395/200.06
5579535	November 1996	Orlen et al.	
5581595	December 1996	Iwashita et al.	379/57
5600703	February 1997	Dang et al.	455/31.3
5608786	March 1997	Gordan	
5623605	April 1997	Keshav et al.	
5638450	June 1997	Robson	455/31.3
5649289	July 1997	Wang et al.	455/31.3
5671354	September 1997	Ito et al.	395/187.01
5673322	September 1997	Pepe et al.	
5675507	October 1997	Bobo, II	364/514R
5708780	January 1998	Levergood et al.	395/200.12
5708828	January 1998	Coleman	
5727159	March 1998	Kikinis	
5740252	April 1998	Minor et al.	
5742668	April 1998	Pepe et al.	455/57
5742905	April 1998	Pepe et al.	455/461
5745706	April 1998	Wolfberg et al.	
5751798	May 1998	Mumick et al.	
5809415	September 1998	Rossmann	
5812768	September 1998	Page et al.	
5841764	November 1998	Roderique et al.	
5852717	December 1998	Bhide et al.	
5854936	December 1998	Pickett	395/710
5867153	February 1999	Grandcolas et al.	
5884284	March 1999	Peters et al.	
5898904	April 1999	Wang	455/31.3
5909485	June 1999	Martin et al.	
5918013	June 1999	Mighdoll et al.	
5940589	August 1999	Donovan et al.	
5943046	August 1999	Cave et al.	455/5.1
5943399	August 1999	Bannister et al.	
5958006	September 1999	Eggleston et al.	
5995606	November 1999	Civanlar et al.	
5999827	December 1999	Sudo et al.	455/566
6009413	December 1999	Webber et al.	705/26
6009422	December 1999	Ciccarelli	

h e b b g e e e f e e ef b e

6021437	February 2000	Chen et al.
6023698	February 2000	Lavey, Jr. et al.
6031904	February 2000	An et al.
6035189	March 2000	Ali-Vehmas et al.
6049711	April 2000	Ben-Yehezkel et al.
6049821	April 2000	Theriault et al.
6055441	April 2000	Wieand et al.
6058422	May 2000	Ayanoglu et al.
6065120	May 2000	Laursen et al.
6085105	July 2000	Becher
6108554	August 2000	Kawamoto
6119137	September 2000	Smith et al.
6119155	September 2000	Rossmann et al.
6122403	September 2000	Rhoades
6131067	October 2000	Girerd et al.
6138158	October 2000	Boyle et al.
6157823	December 2000	Fougnies et al.
6161140	December 2000	Moriya
6167253	December 2000	Farris et al.
6185184	February 2001	Mattaway et al.
6195693	February 2001	Berry et al.
6233608	May 2001	Laursen et al.
6237031	May 2001	Knauerhase et al.
6247048	June 2001	Greer et al.

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0 646 856	April 1995	EP	
0691619	January 1996	EP	
0 646 856	December 1996	EP	
0691619	June 1997	EP	
0812120	December 1997	EP	
0893760	January 1999	EP	
0812120	May 1999	EP	
59-41047	March 1984	JP	
5-35421	February 1993	JP	
5-233191	September 1993	JP	
6-110637	April 1994	JP	
6-175764	June 1994	JP	
7-13671	January 1995	JP	
7-263187	October 1995	JP	
WO 93/16550	August 1993	WO	
WO 93/16550	August 1993	WO	
WO 97/14244	April 1997	WO	
WO 97/27546	July 1997	WO	

h e b b g e e e f e e ef b e

WO 97/41654 November 1997 WO

OTHER PUBLICATIONS

GloMop Group, "GloMop: Global Mobile Computing By Proxy", pp. 1-12, Sep. 13, 1995, University of California, Berkeley, California, XP-002094009.
S.S. Chakraborty, "Mobile Multimedia: In Context to ATM Transport and GSM/GPRS Mobile Access Networks", pp. 1937-1941, Helsinki University of Technology, Espoo, Finland, 1995.
HDTP Specification, Version 1.1-Draft, pp. 1-40, Redwood Shores, CA, Unwired Planet, Inc., Jul. 15, 1997.
HDML 2.0 Language Reference, Version 2.0, pp. 1-56, Redwood Shores, CA, Unwired Planet, Inc., Jul. 1997.
Japanese Publication, "Nifty Serve", pp. 14, 34, 39, and 58-159 Oct. 5, 1992.
Japanese Article, "Mosaic", Unix Magazine, pp. 36-44, Mar. 1994.
Japanese Publication, "Introduction to HTML-WWW Publishing", pp. 9, 12, 14, 15, 20, 32-34, 162-163, 269, 286, 287, and 270, Jun. 30, 1995.
Japanese Publication, Larry Aronson, "Introduction to HTML", Jul. 1, 1995.
Nekkei Multimedia Magazine, pp. 109-111, Nov. 1995.
Mac Power Magazine, pp. 105 and 268, Jun. 1995.
M. Liljeberg et al., "Optimizing World-Wide Web for Weakly Connected Mobile Workstations: An Indirect Approach" International Workshop on Services in Distributed and Networked Environments, pp. 132-139, Jun. 5, 1995, XP000764774.
M.F. Kaashoek et al., "Dynamic Documents: Mobile Wireless Access to the WWW", pp. 179-184, Dec. 8, 1994, XP0002016896.
S. Gessler et al., "PDAs as mobile WWW Browsers", Computer Networks and ISDN System, vol. 28, No. 1, pp. 53-59, Dec. 1, 1995, XP004001210.
G.M. Voelker et al., "Mobisaic: An Information System for a Mobile Wireless Computing Enviroment", vol. 5, No. 10, pp. 185-190, Jan. 1, 1995, XP002062595.
C. Erlandson et al., "WAP-The Wireless Application Protocol" Ericsson Review No. 4, pp. 150-153, Stockholm, 1998, XP-000792053.
M. F. Kaashoek et al.: "Dynamic Documents: Mobile Wireless Access to the WWW" Proceedings, Workshop on Mobile Computing Systems and Applications, Dec. 8, 1994, pp. 179-184, XP0002016896.
G. M. Voelker et al.: "Mobisaic: An Information System for a Mobile Wireless Computing Environment" Proceedings, Workshop on Mobile computing Systems and Applications, vol. 5, No. 10, Jan. 1, 1995, pp. 185-190, XP002062595.

ART-UNIT: 2683

PRIMARY-EXAMINER: Trost; William

ASSISTANT-EXAMINER: Nguyen; Simon

ATTY-AGENT-FIRM: Blakely, Sokoloff, Taylor & Zafman LLP

ABSTRACT:

A two-way data communication device such as a data ready cellular telephone, a two-way pager, or a telephone communicates via a two-way data communication network with a server computer on a computer network that has an interface to the two-way data communication network, i.e, is coupled to the two-way data communication network. For example, the computer network can be a corporate wide area network, a corporate local-area network, the Internet, or any combination of computer networks. The two-way data communication device utilizes a client module to transmit message including a resource selector chosen by the user to a server on a server computer on the computer network. The server processes the message and transmits a response over the two-way data communication network to the client

module. The client module interprets the response and presents the response to the user via a structured user interface. Alternatively, the user transmits a request that directs the server to transmit the response to the request to another location or to another user.

30 Claims, 56 Drawing figures

☐ 3. Document ID: US 6578069 B1

L11: Entry 3 of 27 File: USPT Jun 10, 2003

US-PAT-NO: 6578069
DOCUMENT-IDENTIFIER: US 6578069 B1

TITLE: Method, data structure, and computer program product for identifying a network resource

DATE-ISSUED: June 10, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hopmann; Alexander I.	Seattle	WA		
Anderson; Rebecca L.	Redmond	WA		
Deen; Brian J.	North Bend	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Microsoft Corporation	Redmond	WA			02

APPL-NO: 09/ 412071 [PALM]
DATE FILED: October 4, 1999

INT-CL: [07] G06 F 12/00

US-CL-ISSUED: 709/203; 707/203, 707/201
US-CL-CURRENT: 709/203; 707/201, 707/203

FIELD-OF-SEARCH: 707/101, 707/10, 707/202, 707/201, 707/8, 707/203, 714/12, 714/15, 714/4, 709/226, 709/220, 709/223, 709/205, 710/302, 713/100, 713/200, 713/201, 711/144, 711/141

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5560005	September 1996	Hoover et al.	707/10
5600834	February 1997	Howard	707/201
5812773	September 1998	Norin	709/201

h e b b g e e e f e e ef b e

5812793	September 1998	Shakib et al.	707/201
5884325	March 1999	Bauer et al.	705/40
6049799	April 2000	Mangat et al.	370/396
6182117	January 2001	Christie et al.	709/205
6256740	July 2001	Muller et al.	713/201

ART-UNIT: 2154

PRIMARY-EXAMINER: An; Meng-Al T.

ASSISTANT-EXAMINER: Lin; Kenny

ATTY-AGENT-FIRM: Workman, Nydegger & Seeley

ABSTRACT:

Techniques are presented for allowing clients and servers in a computer network executing the WebDAV protocol to identify a specific version of a specific resource a specific version using a resource tag. The resource can be identified even though it has been changed at a server or at an off line local cache of a client that is disconnected from the network and then later re connected to the network for uploading. Also, a resource UID is presented that will not change despite changes to the URL or the resource tag of the resource. Each resource UID of each resource can be cached locally at a client and can be stored at network server in an index. The index allows the resource to be identified uniquely across a collection in a database at a server, across a database at the server, across the entire server, or across all servers in the network.

26 Claims, 11 Drawing figures

☐ 4. Document ID: US 6578054 B1

L11: Entry 4 of 27 File: USPT Jun 10, 2003

US-PAT-NO: 6578054
DOCUMENT-IDENTIFIER: US 6578054 B1

TITLE: Method and system for supporting off-line mode of operation and synchronization using resource state information

DATE-ISSUED: June 10, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hopmann; Alexander I.	Seattle	WA		
Anderson; Rebecca L.	Redmond	WA		
Deen; Brian J.	North Bend	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE

h e b b g e e e f e e ef b e

Microsoft Corporation Redmond WA 02

APPL-NO: 09/ 412766 [PALM]
DATE FILED: October 4, 1999

INT-CL: [07] G06 F 12/00

US-CL-ISSUED: 707/201; 707/203, 709/203
US-CL-CURRENT: 707/201; 707/203, 709/203

FIELD-OF-SEARCH: 707/201, 707/202, 707/203, 707/10, 707/8, 709/205, 709/201, 709/203, 709/204, 709/217, 711/161, 711/141, 711/144

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5600834	February 1997	Howard	395/617
5737601	April 1998	Jain et al.	395/617
5787262	July 1998	Shakib et al.	707/201
5812773	September 1998	Norin	395/200.34
5812793	September 1998	Shakib et al.	395/200.31
5884325	March 1999	Bauer et al.	705/40
5884328	March 1999	Mosher, Jr.	707/202
5924094	July 1999	Sutter	707/1
5924096	July 1999	Draper et al.	707/10
5991771	November 1999	Falls et al.	707/201
6058401	May 2000	Stamos et al.	707/10
6405218	June 2002	Boothby	707/201

OTHER PUBLICATIONS

Martin, J., "Design and Strategy for Distributed Data Processing", Prentice-Hall, 1998, pp. 272-304.*
Borenstein, et al., RFC 1521, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies," Sep. 1993.
Fielding, et al., RFC 2068, "Hypertext Transfer Protocol--HTTP/1.1," Jan. 1997.
Slein, et al., RFC 2291, "Requirements for a Distributed Authoring and Versioning Protocol For the World Wide Web," Feb. 1998.
Goland, et al., RFC 2518, "HTTP Extensions for Distributed Authoring--WEBDAV," Feb. 1999.
Fielding, et al., RFC 2316, "Hypertext Transfer Protocol--HTTP/1.1," Jun. 1999.
Yavin, D., "Replication's Fast Track,"BYTE, Aug. 1995, pp. 88a-88d, 90.

ART-UNIT: 2154

PRIMARY-EXAMINER: An; Meng-Al T.

ASSISTANT-EXAMINER: Lin; Kenny

ATTY-AGENT-FIRM: Workman, Nydegger & Seeley

h e b b g e e e f e e ef b e

ABSTRACT:

Systems and methods for synchronizing multiple copies of data in a network environment that includes servers and clients so that incremental changes made to one copy of the data can be identified, transferred, and incorporated into all other copies of the data. The synchronization can be accomplished regardless of whether modifications to the data have been made by a client while the client is in an on-line or off-line mode of operation. The clients cache data locally as data are modified and downloaded. The caching enables the clients to access the data and allows the synchronization so be performed without transmitting a particular version more than once between a client and a server. Such elimination of redundant data transmission results in an efficient use of time and network bandwidth.

20 Claims, 9 Drawing figures

☐ 5. Document ID: US 6499031 B1

L11: Entry 5 of 27 File: USPT Dec 24, 2002

US-PAT-NO: 6499031
DOCUMENT-IDENTIFIER: US 6499031 B1
**** See image for Certificate of Correction ****

TITLE: Systems and methods for using locks with computer resources

DATE-ISSUED: December 24, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hopmann; Alexander I.	Seattle	WA		
Van; Van C.	Kirkland	WA		
Deen; Brian J.	North Bend	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Microsoft Corporation	Redmond	WA			02

APPL-NO: 09/ 360753 [PALM]
DATE FILED: July 26, 1999

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/8; 707/9, 709/104, 710/36, 710/220, 710/240
US-CL-CURRENT: 707/8; 707/9, 710/220, 710/240, 710/36, 718/104

FIELD-OF-SEARCH: 707/8, 707/9, 707/1-6, 707/10, 713/200-202, 709/217-219, 709/104-107, 380/25, 710/36, 710/200, 710/220, 710/240-244

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

h e b b g e e e f e e ef b e

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5774551	June 1998	Wu et al.	380/25
5832484	November 1998	Sankaran et al.	707/8
5867494	February 1999	Krishnaswamy et al.	370/352
5889952	March 1999	Hunnicutt et al.	709/219
6065117	May 2000	White	713/159
6160547	December 2000	Roth	345/328

ART-UNIT: 2172

PRIMARY-EXAMINER: Alam; Hosain T.

ATTY-AGENT-FIRM: Workman, Nydegger & Seeley

ABSTRACT:

Provided is a method for locking computer resources and for accessing locked computer resources. Resources being used by remote users can be locked such that other remote users and local users have restricted access to those resources. The remote user provides the resource to be locked, the type of lock to place on the resource and the duration of the lock. If the resource is available and the user has the proper credentials and the proper access permissions, the resource is locked and a lock token is provided to the remote user. A resource handle is also provided to the remote user. Once a resource has been locked, the lock token must be provided and verified before access to the resource is granted. Because the locks can be discovered, a security token of the owner of the lock is associated with the lock token. The security token of the remote user must also match the security token associated with the stored lock token before access to the resource is granted. The locks can either expire or be refreshed. If a lock expires, it is removed from memory and the resource handle is released. Also, particular user agents are given a fixed lock token in order to access system resources. These particular agents view the resource as locked, but the non-unique fixed lock token indicates to the system that the resource is not in fact locked. Other particular agents are given an extended timeout period to account for real time latencies of the computer system.

36 Claims, 4 Drawing figures

☐ 6. Document ID: US 6430409 B1

L11: Entry 6 of 27 File: USPT Aug 6, 2002

US-PAT-NO: 6430409
DOCUMENT-IDENTIFIER: US 6430409 B1

TITLE: Method and architecture for an interactive two-way data communication network

DATE-ISSUED: August 6, 2002

h e b b g e e e f e e ef b e

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Rossmann; Alain	Menlo Park	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Openwave Systems Inc.	Redwood City	CA			02

APPL-NO: 08/ 978701 [PALM]
DATE FILED: November 26, 1997

PARENT-CASE:
This application is a continuation of application Ser. No. 08/570,210, filed Dec. 11, 1995.

INT-CL: [07] H04 Q 7/20, H04 Q 7/32, H04 Q 7/28

US-CL-ISSUED: 455/422; 455/426, 455/564, 455/565, 455/466, 455/552, 455/517, 709/203, 709/217, 370/230, 370/352, 370/328
US-CL-CURRENT: 455/422.1; 370/230, 370/328, 370/352, 455/426.1, 455/466, 455/517, 455/552.1, 455/564, 455/565, 709/203, 709/217

FIELD-OF-SEARCH: 455/426, 455/3.05, 455/550, 455/564, 455/565, 455/517, 455/412, 455/413, 455/414, 455/419, 455/466, 455/556, 455/557, 455/552, 455/422, 709/203, 709/224, 709/246, 709/228, 709/236, 709/217, 370/230, 370/352, 370/328

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4787028	November 1988	Finfrock et al.	
4812843	March 1989	Champion, III et al.	
5008925	April 1991	Pireh	
5335276	August 1994	Thompson et al.	
5465401	November 1995	Thompson	
5491605	February 1996	Hughbanks et al.	
5548636	August 1996	Bannister et al.	
5555446	September 1996	Jasinski	
5560008	September 1996	Johnson et al.	
5577100	November 1996	McGregor et al.	
5577103	November 1996	Foti	
5577209	November 1996	Boyle et al.	
5579535	November 1996	Orlen et al.	
5606786	March 1997	Presby	
5625605	April 1997	Sullivan et al.	
5671354	September 1997	Ito et al.	
5708828	January 1998	Coleman	
5727159	March 1998	Kikinis	
5740252	April 1998	Minor et al.	455/412

h e b b g e e e f e e ef b e

5742668	April 1998	Pepe et al.	
5745706	April 1998	Wolfberg et al.	
5751798	May 1998	Mumick et al.	
5854936	December 1998	Pickett	
5867153	February 1999	Grandcolas et al.	
5884284	March 1999	Peters et al.	
5909485	June 1999	Martin et al.	
5918023	June 1999	Mighdoll et al.	
5940589	August 1999	Donovan et al.	
5958006	September 1999	Eggleston et al.	
6049711	April 2000	Ben-Yehezkel et al.	
6085105	July 2000	Becher	
6119137	September 2000	Smith et al.	
6122403	September 2000	Rhoads	709/203
6157823	December 2000	Fougnies et al.	
6167253	December 2000	Farris et al.	455/414
6185184	February 2001	Mattaway et al.	709/203

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0812120	December 1997	EP	
0812120	December 1997	EP	
WO 97/41654	November 1997	WO	

OTHER PUBLICATIONS

HDTP Specification, Version 1.1-Draft, pp. 1-40, Redwood Shores, CA, Unwired Planet, Inc., Jul. 15, 1997.
HDML 2.0 Language Reference, Version 2.0, pp. 1-56, Redwood Shores, CA, Unwired Planet, Inc., Jul. 1997.

ART-UNIT: 2683

PRIMARY-EXAMINER: Trost; William

ASSISTANT-EXAMINER: Ferguson; Keith

ATTY-AGENT-FIRM: Blakely, Sokoloff, Taylor & Zafman LLP

ABSTRACT:

A two-way data communication device such as a data ready cellular telephone, a two-way pager, or a telephone communicates via a two-way data communication network with a server computer on a computer network that has an interface to the two-way data communication network, i.e, is coupled to the two-way data communication network. For example, the computer network can be a corporate wide area network, a corporate local area network, the Internet, or any combination of computer networks. The two-way data communication device utilizes a client module to transmit message including a resource selector chosen by the user to a server on a server computer on the computer network. The server processes the message and

h e b b g e e e f e e ef b e

transmits a response over the two-way data communication network to the client module. The client module interprets the response and presents the response to the user via a structured user interface. Alternatively, the user transmits a request that directs the server to transmit the response to the request to another location or to another user.

93 Claims, 56 Drawing figures

☐ 7. Document ID: US 6385595 B1

L11: Entry 7 of 27 File: USPT May 7, 2002

US-PAT-NO: 6385595
DOCUMENT-IDENTIFIER: US 6385595 B1

TITLE: Electronic statement presentment system

DATE-ISSUED: May 7, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Kolling; Ray	Menlo Park	CA		
Occhino; Michael	Castro Valley	CA		
Roughgarden; Jeffrey D.	Redwood City	CA		
Hayward; James T.	Shawnigan Lake			CA

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Visa International Service Association	Foster City	CA			02

APPL-NO: 09/ 344829 [PALM]
DATE FILED: June 25, 1999

PARENT-CASE:
This application is a continuation of U.S. patent application No. 08/947,629, filed Oct. 8, 1997 now U.S. Pat. No. 5,963,925, which in turn claims priority of U.S. provisional patent application No. 60/028,095, filed Oct. 9, 1996, entitled "Electronic Statement Presentation", by inventors Kolling, et al., which is hereby incorporated by reference in its entirety for all purposes.

INT-CL: [07] G06 F 17/60

US-CL-ISSUED: 705/40; 705/34
US-CL-CURRENT: 705/40; 705/34

FIELD-OF-SEARCH: 705/30, 705/34, 705/35, 705/39, 705/40, 705/42, 705/44, 705/45

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

h e b b g e e e f e e ef b e

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4634845	January 1987	Hale et al.	
4689478	August 1987	Hale et al.	
4713761	December 1987	Sharpe et al.	
4713837	December 1987	Gordon	
4799156	January 1989	Shavit et al.	
4823264	April 1989	Deming	
4949272	August 1990	Vanourek et al.	
4988849	January 1991	Sasaki et al.	
5007084	April 1991	Materna et al.	
5025373	June 1991	Keyser, Jr. et al.	
5193055	March 1993	Brown et al.	
5193110	March 1993	Jones et al.	
5220501	June 1993	Lawlor et al.	
5237159	August 1993	Stephens et al.	
5262942	November 1993	Earle	
5265033	November 1993	Vajk et al.	
5283829	February 1994	Anderson	
5287270	February 1994	Hardy et al.	
5325290	June 1994	Cauffman et al.	
5336870	August 1994	Hughes et al.	
5367561	November 1994	Adler et al.	
5373561	December 1994	Haber et al.	
5383113	January 1995	Kight et al.	
5410598	April 1995	Shear	
5420405	May 1995	Chasek	
5426594	June 1995	Wright et al.	
5453601	September 1995	Rosen	
5455407	October 1995	Rosen	
5465206	November 1995	Hilt et al.	
5473143	December 1995	Vak et al.	
5483445	January 1996	Pickering	
5484988	January 1996	Hills et al.	
5485370	January 1996	Moss et al.	
5496991	March 1996	Delfer, III et al.	
5508817	April 1996	Kunigami	
5517569	May 1996	Clark	
5532464	July 1996	Josephson et al.	
5557518	September 1996	Rosen	
5699528	December 1997	Hogan	705/40
5761650	June 1998	Munsil	705/34
5832460	November 1998	Bednar et al.	705/27

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0745947	April 1996	JP	

OTHER PUBLICATIONS

Epper "Checkfree to test on-line payment with utility companies in 3 states"; Oct. 1995, American Banker, v160, n218, p14(1), Dialog file 148, Accession No. 08297809.*
Michelle; "US order's stock price up sharply rise linked to news of internet product"; Wahington Post; Dialog file 146, Accession No. 4044098, Dec. 1995.*
Shapiro; "Getting, staying in contact which disclose enclosure and a plurality of templates"; MacWeek, v4, N33, p59(5); Dialog file 148, Accession No. 04810289, Oct. 1990.*
"New product enables merchants to present bills on internet"; PR News, p1204DCM016; Dialog file 16, Accession No. 04105573, Dec. 1995.*
Cary; "Document assembly using WordPerfect and word", Jul. 1994; Information Today v11n7 pp. 10-11; Proquest file 00907550.*
Jim; "Electronic Billing Via Internet" Dec. 21, 1995; Newsbytes News Network, Stillwater.*
"Online Fund System Among Main Banks", Oct. 1989, Money Circulation System.
"Money circulation Data System Center", Electronic Banking, Oct. 31, 1986, pp. 35-41.
D.W. Davies and W.W. Price, translated by Tadahiro Ueno, "Electronic Money Movement and Intelligent Token", Dec. 5, 1985, Network Security, Chapter 10, pp. 251-263 and pp. 283-287.
"Notice of the Reason for Refusal", Mar. 31, 1998, Japanese Patent Office.
Dr. Denis Manelski, Bill & Pay Minitel, presented on Apr. 22, 1992.
Announcement released on Mar. 20, 1995, on web site http://ganges.cs.tcd.ie/mpeirce/Project/Press/fboi.html.
David Chaum, World's finest electronic cash payment over computer networks, May 27, 1994, on web site http://ganges.cs.tcd.ie/mepeirce/Project/press/ec_press.html.
Eric Mankin, the Check is in the E-Mail (And, Coming Soon, Electronic Greenbacks), Nov. 7, 1994, USC Chronicle.
Presented by Dennis J. Pope of Manufacturing Hanover Trust Company.
Presented by Paul J. Mila of Online Resources and Communications Corp.
Chris Shipley, I three away my checkbook, Nov. 1990, PC Computing.
George C. White, Have you heard? "Check and list" is obsolete for receiving consumer bill payments. Sep./Oct. 1990, Journal of Cash Management.
"International Search Report", Mar. 23, 1998, European Patent Office, 3 pages.
Tanja Lian, "Apply Your Marketing Talent to promote On-line Banking", May 1, 1996, Bank Marketing, pp. 25-30.
Sirbu, et al., "NetBill: An Internet Commerce System Optimized for Network Delivered Services", Mar. 1995, IEEE.
David Jones, "US Banks Experiment with Home Banking", Jan. 1984, The Banker.
Debbie Gunthrie Haer, "Two-way Cable TV to provide Home Banking in Omaha", Feb. 15, 1982, Bank News.
Garrett P. Bromley, "At the Podium A Bankers View: Prescribing Financial Relief for New Doctors", Sep. 1982, United States Banker.
Michael P. Sullivan, "Financial Forum: the HBI Role in the Home Banking Revolution", United States Banker.
Joan Prevete Hyman, "Spotlight on software--Switch Software to Take on POS, Home Banking Functions: These packages Will be Able to Handle More Than Just ATM messages, Say Vendors", Feb. 1983, Bank Systems & Equipement.
Lisa Fickenscher, "Credit/Debit/ATMs: Online Resources' Home Banking Patent Hits Hot Buttons Throughout Industry", Feb. 17, 1994, American Banker.
"Finance, Next in Banking": Pay Bills by Phone.
Robert M. Garsson, "NBD Offers Electronic Highway for Network of Shared ATMs", Operations, Technology.
"Special Report: Money Goes Electronic in the 1970's", Jan. 13, 1968, Business

Week.
John P. Fisher, "In-Home Banking Today and Tomorrow", Jun. 1982, Journal of Retail Banking, vol. IV, No. 2.
Robert Trigaux, "Home Banking reaches Critical Juncture: Principal providers May Already Be in Place; Small-Bank Options Emerge", Oct. 19, 1982, American Banker, vol. 147, Issue 204.
Jerry Adams, "Home Banking Interchange offers More than just Credits and Debits", Jul. 6, 1983, American Banker, vol. 148, Issue 130.
"Home Banking Interchange: New Venture to Test Home Banking Acceptance", May 27, 1983, Revolving Credit & Electronic Systems Letter.
Michael P. Sullivan, "Home-Banking: The Ultimate Delivery System", The Bankers Magazine.
John A. Farnsworth, Technology Report: Home Banking--Parts of a Bigger Picture, Jun. 1983, United States Banker.
"Electronic Home Banking Lets Customers Pay bills Around the Clock", Apr. 1984, The Magazine of Bank Administration.
Tekla S. Perry, "Electronic Banking Goes to market", Feb. 1998, IEEE Spectrum.
Edward J. Hogan, "EFT Technology--Present and Future", Dec. 16, 1976, National Commission of Electronic Fund Transfer, Suppliers Committee Public Hearing (2 Volumes).
Maria Osborn Howard, "Crestar to Test At-Home Banking", Feb. 26, 1994, Richmond Times-Dispatch.
David O. Tyson, "Banks in Denver, San Francisco to Offer Customers PC Links", Oct. 23, 1985.
Anderson et al., "An Electronic Cash and Credit System", 1966, American Management Association.
"Base I: A Real-Time System for Interchange Authorizations", VISA U.S.A, Inc.
"Base II: An Electronic System for Worldwide Interchange", VISA U.S.A., Inc.
Lynch, et al., "Digital Money, The New Era of Internet Commerce", 1996, John Wiley & Sons, Inc.
Tekla S. Perry, "Electronic Money: Toward a Virtual Wallet", IEEE Spectrum, Feb. 1997.
Edward W. Kelley, Jr., "The Future of Electronic Money: A Regulator's Perspective", IEEE Spectrum, Feb. 1997.
Marvin A. Sirbu, "Credit and Debits on the Internet", IEEE Spectrum Feb. 1997.
David Chaum, et al., "Minting Electronic Cash", IEEE Spectrum Feb. 1997.
Peter S. Gemmell, "Traceable E-Cash", IEEE Spectrum Feb. 1997.
Stanley E. Morris, "Crime and Prevention: A Treasury Viewpoint", IEEE Spectrum Feb. 1997.
Baldwin et al., "Locking the E-Safe", IEEE Spectrum Feb. 1997.
Carol Hovenga Fancher, "In Your Pocket: Smartcards", IEEE Spectrum Feb. 1997.
Michael C. McChesney, "Banking in Cyberspace: An Investment in Itself", IEEE Spectrum 1997.
Steven M.H. Wallman, "Technology Takes to Securities Trading", IEEE Spectrum 1997.
Alfred R. Berkeley, III, "Nasdaq's Technology Floor: Its President Takes Stock", IEEE Spectrum 1997.
Mike TerMaat, "The Economics of E-Cash", IEEE Spectrum 1997.
Howard Anderson, "Money and the Internet: A Strange New Relationship" IEEE Spectrum 1997.
"To Probe Further", IEEE Spectrum 1997.
CyberCash's Secure Internet Payment Services, CyberCash, Inc., Reston, Virginia 22091.
Leslie Marable, "A test Moves net-Based Bill Payment a Step Closer", WebWeek, vol. three, Issue Three, Feb. 3, 1997.
"Visa ePay: A Fully Electronic Solution for Biller Payment Delivery", Visa U.S.A Inc. V4068-06-86, 1996.

ART-UNIT: 2163

PRIMARY-EXAMINER: Choi; Kyle J.

h e b b g e e e f e e ef b e

ASSISTANT-EXAMINER: Jeanty; Romain

ATTY-AGENT-FIRM: Beyer Weaver & Thomas, LLP

ABSTRACT:

An electronic statement presentment (ESP) system replaces the preparation and mailing of paper statements and invoices from a biller with electronic delivery. Electronic statements have the same look as paper statements as well as including video, audio, graphics, and custom enclosures. Statements are segmented into mandatory and optional components to minimize download time. The ESP system operates independently or is an enhancement to any suitable electronic bill payment system. A central switch computer coordinates template storage, validation, routing and message passing between billers, workstations and consumer financial institutions (CFI). A template authoring workstation (TAWS) creates a template of static biller information to serve as a basis for the electronic statement. The template is stored in a template library at the switch. The switch validates the template by sending it to a template validation workstation (TVAL). Batches of customer statement data are sent from a biller's legacy invoicing system to a statement origination workstation (SORG) along with a template identifier.

21 Claims, 20 Drawing figures

☐ 8. Document ID: US 6345244 B1

L11: Entry 8 of 27 File: USPT Feb 5, 2002

US-PAT-NO: 6345244
DOCUMENT-IDENTIFIER: US 6345244 B1

TITLE: System, method, and product for dynamically aligning translations in a translation-memory system

DATE-ISSUED: February 5, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Clark; Jonathan P.	Ashland	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Lionbridge Technologies, Inc.	Waltham	MA			02

APPL-NO: 09/ 085471 [PALM]
DATE FILED: May 27, 1998

INT-CL: [07] G06 F 17/28

US-CL-ISSUED: 704/2; 704/7
US-CL-CURRENT: 704/2; 704/7

h e b b g e e e f e e ef b e

FIELD-OF-SEARCH: 704/2-8, 704/9-10, 704/275, 704/257, 704/531, 707/530, 707/536

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5510981	April 1996	Berger et al.	704/2
5677835	October 1997	Crbonell et al.	704/9
5724593	March 1998	Hargrave, III et al.	704/2
5787386	July 1998	Kaplan et al.	704/9
5805832	September 1998	Brown et al.	704/2
5826220	October 1998	Takeda et al.	704/2
5850561	December 1998	Church et al.	704/10
5867811	February 1999	O'Donoghue	704/10
5893134	April 1999	O'Donoghue et al.	704/2
5907821	May 1999	Kaji et al.	704/2

ART-UNIT: 2644

PRIMARY-EXAMINER: Edouard; Patrick N.

ATTY-AGENT-FIRM: Wolf, Greenfield & Sacks, P.C.

ABSTRACT:

A computer-implemented system, method, and product associate words, phrases, or other characters ("words") to be translated with previous translations of such words, if a previous translation exists. The words to be translated, and their identifying attributes (such as, for example, formatting information), are extracted from one or more source files. The previous translations, and their identifying attributes, are extracted from one or more target files. If a previous translation does not exist, the words to be translated, and their attributes, are copied to create a pseudo-previous translation, with attributes. Each word to be translated is associated with a corresponding previous translation based upon commonalities of their attributes. Such words may also be associated based upon their respective locations in their respective files. Each word to be translated, and its attributes, may be stored with a previous translation in a source-target pair record of a source-target pair database. Each source-target pair record may include a propagation flag identifying whether the previous translation stored in the source-target pair record is to be propagated to other occurrences of the associated word to be translated in the source-target pair database. Each source-target pair record may also include a pointer to a page of an occurrence book having pages, wherein each page includes pointers to a common word to be translated in records of the source-target pair database.

68 Claims, 21 Drawing figures

☐ 9. Document ID: US 6345243 B1

h e b b g e e e f e e ef b e

L11: Entry 9 of 27 File: USPT Feb 5, 2002

US-PAT-NO: 6345243
DOCUMENT-IDENTIFIER: US 6345243 B1

TITLE: System, method, and product for dynamically propagating translations in a translation-memory system

DATE-ISSUED: February 5, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Clark; Jonathan P.	Ashland	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Lionbridge Technologies, Inc.	Waltham	MA			02

APPL-NO: 09/ 085468 [PALM]
DATE FILED: May 27, 1998

PARENT-CASE:
RELATED APPLICATION The following application is related to the present application. U.S. patent application Ser. No. 09/085,471, entitled "SYSTEM, METHOD, AND PRODUCT FOR DYNAMICALLY ALIGNING TRANSLATIONS IN A TRANSLATION-MEMORY SYSTEM,", naming as inventor Jonathan Clark, assigned the assignee of the present invention and filed concurrently herewith.

INT-CL: [07] G06 F 17/28

US-CL-ISSUED: 704/2; 704/7
US-CL-CURRENT: 704/2; 704/7

FIELD-OF-SEARCH: 704/2-8, 704/10, 707/530, 707/536

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5477450	December 1995	Takeda et al.	704/2
5510981	April 1996	Berger et al.	704/2
5528491	June 1996	Kuno et al.	704/9
5677835	October 1997	Crbonell et al.	704/9
5724593	March 1998	Hargrave, III et al.	704/2
5787386	July 1998	Kaplan et al.	704/9
5805832	September 1998	Brown et al.	704/2
5826220	October 1998	Takeda et al.	704/2
5850561	December 1998	Church et al.	704/10
5867811	February 1999	O'Donoghue	704/10
5893134	April 1999	O'Donoghue et al.	704/2

5907821 May 1999 Kaji et al. 704/2

ART-UNIT: 2644

PRIMARY-EXAMINER: Edouard; Patrick N.

ATTY-AGENT-FIRM: Wolf, Greenfield & Sacks, P.C.

ABSTRACT:

A computer-implemented system, method, and product are provided to propagate an externally produced translation of a first occurrence of a word, phrase, or other group of characters to be translated (referred to herein as a "translatable source segment") in a first source file to at least one occurrence of a corresponding target segment in one or more target files of a target project. The corresponding target segment corresponds with a second occurrence of the translatable source segment in any of the one or more source files. The source files may have the same file format, or two or more of the source files may have different file formats. Also, two or more source files may have character formats that are different from each other, and/or they may be written in different languages. In one implementation, the propagator propagates the externally produced translation to all corresponding target segments in the target project that correspond with the translatable source segment. Also, the propagator may propagate the externally produced translation to user-selected corresponding target segments in the target project that corresponds with the translatable source segment. The externally produced language translation may be generated by a human translator, or by a machine translation tool. In one embodiment, the externally produced language translation is derived from a legacy file. Also, the externally produced language translation may be derived from a corresponding target segment that corresponds with the translatable source segment.

59 Claims, 20 Drawing figures

☐ 10. Document ID: US 6249795 B1

L11: Entry 10 of 27 File: USPT Jun 19, 2001

US-PAT-NO: 6249795
DOCUMENT-IDENTIFIER: US 6249795 B1

TITLE: Personalizing the display of changes to records in an on-line repository

DATE-ISSUED: June 19, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Douglis; Frederick	Somerset	NJ		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
AT&T Corp.	New York	NY			02

h e b b g e e e f e e ef b e

APPL-NO: 08/ 965060 [PALM]
DATE FILED: November 5, 1997

PARENT-CASE:
This application is a continuation-in-part of the U.S. application Ser. No. 08/797,756, which was filed Feb. 7, 1997, issued as U.S. Pat. No. 5,860,071. This is a continuation-in-part of U.S. application Ser. No. 08/549,359 which was filed Oct. 27, 1995, which is herein incorporated by reference.

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/511; 707/530
US-CL-CURRENT: 715/511; 715/530

FIELD-OF-SEARCH: 707/500, 707/501, 707/502, 707/503, 707/504, 707/505, 707/506, 707/507, 707/508, 707/511, 707/530, 707/103

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4807182	February 1989	Queen	364/900
5008853	April 1991	Bly et al.	364/900
5325478	June 1994	Shelton	395/148
5438661	August 1995	Ogawa	395/157
5860071	January 1999	Ball et al.	707/100
5878213	March 1999	Bittinger et al.	395/200.33
5894585	April 1999	Inoue	395/827
5903897	March 1999	Carrier et al.	707/203
5933811	August 1999	Angles et al.	705/14

OTHER PUBLICATIONS

Douglis, Fred et al., "The AT&T Internet Difference Engine: Tracking and Viewing Changes on the Web," AT&T Labs--Research and Lucent Technologies, Bell Laboratories, 1997, pp. 1-33.

ART-UNIT: 216

PRIMARY-EXAMINER: Hong; Stephen S.

ABSTRACT:

The invention provides for customized "what's new" lists for users of shared lists of tracked resources. Identifying information, such as cookies, and state information are used to personalize a shared "what's new" list, so that changes to a document which a particular user has viewed are indicated automatically based on the last time that user viewed the document. Thus, the user is notified of the set of changes that have taken place since the user last viewed a particular set of documents. For any particular document, only the changes which have occurred since the user last viewed that document are used to form the "what's new" list and to view the changes if the user so chooses. Additionally, a user can select multiple

h e b b g e e e f e e ef b e

collections of documents into a single notification system and exclude particular documents or augment the collection to personalize the resources tracked for that user.

26 Claims, 8 Drawing figures

☐ 11. Document ID: US 6237144 B1

L11: Entry 11 of 27 File: USPT May 22, 2001

US-PAT-NO: 6237144
DOCUMENT-IDENTIFIER: US 6237144 B1

TITLE: Use of relational databases for software installation

DATE-ISSUED: May 22, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Delo; John C.	Bellevue	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Microsoft Corporation	Redmond	WA			02

APPL-NO: 09/ 158125 [PALM]
DATE FILED: September 21, 1998

PARENT-CASE:
RELATED APPLICATIONS This application is related to the following applications, all of which are filed on the same day as the present application and are assigned to the same assignee as the present application: "System And Method For Repairing A Damaged Application"--Ser. No. 09/158,126, "Method And System For Restoring A Computer To Its Original State After An Unsuccessful Installation Attempt"--Ser. No. 09/158,124, "A Method For Categorizing And Installing Selected Software Components"--Ser. No. 09/157,695, "System And Method For Managing Locations Of Software Components Via A Source List"--Ser. No. 09/157,974; "Method For Optimizing The Installation Of A Software Product Onto A Target Computer System"--Ser. No. 09/157,853; "Software Installation And Validation Using Custom Actions"--Ser. No. 09/157,776; "Internal Database Validation"--Ser. No. 09/157,828; "Management Of Non-persistent Data In A Persistent Database"--Ser. No. 09/157,883; "Method And System For Advertising Applications"--Ser. No. 09/158,967; "Software Implementation Installer Mechanism"--Ser. No. 09/158,121.

INT-CL: [07] G06 F 9/445

US-CL-ISSUED: 717/11; 709/221, 707/3, 707/10, 707/103Z
US-CL-CURRENT: 717/174; 707/10, 707/103Z, 707/3, 709/221, 717/106, 717/115

FIELD-OF-SEARCH: 717/11, 713/1, 713/100, 709/203, 709/221, 707/10, 707/104, 707/203, 707/3, 707/13R, 707/13Z

h e b b g e e e f e e ef b e

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5361360	November 1994	Ishigami et al.	717/11
5560026	September 1996	Manthuruthil et al.	345/433
5586304	December 1996	Stupek, Jr. et al.	717/11
5835777	November 1998	Staelin	717/11
5859969	January 1999	Oki et al.	709/203
5870611	February 1999	Shrader et al.	717/11
5933647	August 1999	Aronberg et al.	717/11
5950010	September 1999	Hesse et al.	717/11
5960204	September 1999	Yinger et al.	717/11
5974454	October 1999	Apfel et al.	709/221
6006034	December 1999	Heath et al.	717/11
6006035	December 1999	Nabahi	395/712

OTHER PUBLICATIONS

Teri Schiele, "Windows 95 Application Setup Guidelines for Independent Software Vendors", Microsoft Corporation, May 1995, 13 pages.*
Retrieved from the Internet:<URL:
wysiwyg://25/http://msdn.microsoft.com/library/techart/setup.html>.*
Automating Microsoft Transaction Server Client Installation. Product Info. Microsoft Corporation, Jun. 1997 [retrieved on 2000-02-29].*
Retrieved from teh Internet:<URL:
wysiwyg://Main.Prodinfo.6/http://msdn.mi...m/library/backgrnd/html/msdn. sub.13 install.html>.*
Kelly, M., "Gain Control of Application Setup and Maintenance with the New Windows Installer", Microsoft Systems Journal: Sep. 1998, pp. 15-27.

ART-UNIT: 212

PRIMARY-EXAMINER: Trammell; James P.

ASSISTANT-EXAMINER: Dam; Tuan Q.

ATTY-AGENT-FIRM: Kilpatrick Stockton LLP

ABSTRACT:

A method and system for installing computer programs is provided where installation is accomplished based on an "as complete" description of the installed features, components and resources of the computer program. The necessary files and components required for installation of a given feature or component are determined by marking for installation any components which are not presently installed, preparing a script of required installation executions, and then executing the instructions to install the necessary files or components. Components are marked for installation or un-installation in temporary columns and rows which are dynamically added to data tables used to identify components and features which are available for installation. Individual components of a feature may be added or removed by simply marking that component for installation or removal. When the instructions in the installation script are executed, that particular component

will be installed or removed according to the instructions. Installation of a given software application is streamlined because any component of a program application to be installed which is already installed on the user's computer need not be reinstalled upon the installation of the desired feature. Only components or files thereof which must be installed in addition to previously installed components or files need be installed for the installation of the software application program.

24 Claims, 6 Drawing figures

☐ 12. Document ID: US 6065046 A

L11: Entry 12 of 27 File: USPT May 16, 2000

US-PAT-NO: 6065046
DOCUMENT-IDENTIFIER: US 6065046 A

TITLE: Computerized system and associated method of optimally controlled storage and transfer of computer programs on a computer network

DATE-ISSUED: May 16, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Feinberg; Michael A.	Ghent	NY		
Feinberg; Matthew A.	Ghent	NY		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Catharon Productions, Inc.	Ghent	NY			02

APPL-NO: 08/ 902591 [PALM]
DATE FILED: July 29, 1997

INT-CL: [07] G06 F 15/167

US-CL-ISSUED: 709/216; 709/218
US-CL-CURRENT: 709/216; 709/218

FIELD-OF-SEARCH: 395/200.3, 395/200.33, 395/200.46, 395/200.62, 395/200.48, 395/200.49, 395/675, 385/684, 709/200, 709/216, 709/203, 709/232, 709/218, 709/219

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4558413	December 1985	Schmidt et al.	
4954941	September 1990	Redman	
5329619	July 1994	Page et al.	395/200.33

h e b b g e e e f e e ef b e

5341477	August 1994	Pitkin et al.	395/200.56
5388211	February 1995	Hornbuckle	
5426427	June 1995	Chinnock et al.	
5473772	December 1995	Halliwell et al.	
5475813	December 1995	Cieslak et al.	
5475819	December 1995	Miller et al.	395/200.33
5497479	March 1996	Hornbuckle	
5544320	August 1996	Konrad	
5727950	March 1998	Cook et al.	434/350
5732273	March 1998	Srivastava et al.	395/704
5774660	June 1998	Brendel et al.	395/200.31
5819020	October 1998	Beeler, Jr.	395/182.03
5828847	October 1998	Gehr et al.	395/200.69
5850449	December 1998	McManis	380/25

ART-UNIT: 278

PRIMARY-EXAMINER: Maung; Zarni

ATTY-AGENT-FIRM: Sudol; R. Neil Coleman; Henry D.

ABSTRACT:

A computerized system and an associated method for optimally controlling storage and transfer of computer programs between computers on a network to facilitate interactive program usage. In accordance with the method, an applications program is stored in a nonvolatile memory of a first computer as a plurality of individual and independent machine-executable code modules. In response to a request from a second computer transmitted over a network link, the first computer retrieves a selected one of the machine-executable code modules and only that selected code module from the memory and transmits the selected code module over the network link to the second computer.

36 Claims, 29 Drawing figures

☐ 13. Document ID: US 6035121 A

L11: Entry 13 of 27 File: USPT Mar 7, 2000

US-PAT-NO: 6035121
DOCUMENT-IDENTIFIER: US 6035121 A

TITLE: Method and system for localizing a computer program

DATE-ISSUED: March 7, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chiu; Render	Campbell	CA		

h e b b g e e e f e e ef b e

Infantino; Benito Mountain View CA

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY TYPE CODE
Netscape Communication Corporation	Mountain View	CA		02

APPL-NO: 08/ 888859 [PALM]
DATE FILED: July 7, 1997

INT-CL: [07] G06 F 9/45

US-CL-ISSUED: 395/705; 395/709
US-CL-CURRENT: 717/141; 717/136, 717/154, 717/163

FIELD-OF-SEARCH: 395/705, 395/709, 704/8, 704/1, 704/7, 707/536, 345/334

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4706212	November 1987	Toma	704/2
4731735	March 1988	Borgendale et al.	707/4
4870610	September 1989	Belfer	704/2
4916614	April 1990	Kaji et al.	704/2
4953088	August 1990	Suzuki et al.	704/3
5579223	November 1996	Raman	704/1
5583761	December 1996	Chou	707/536
5664206	September 1997	Murow et al.	704/8
5724593	March 1998	Hargrave, III et al.	704/7
5848274	December 1998	Hamby et al.	395/705
5907326	May 1999	Atkin et al.	345/334

OTHER PUBLICATIONS

Windows Developer Spelling Checker, Newsbytes, Jul. 1996.
Discover's O.K. Tax Deal, Credit Card Management, p. 10, Apr. 1995.
The Internet: Top to Bottom, Network Computing, p. 46, May 1996.
INFORMIX: INFORMIX Announces INFORMIX-NEWERA v. 2.0, M2 Presswire, Jul. 1995.

ART-UNIT: 272

PRIMARY-EXAMINER: Hafiz; Tariq R.

ASSISTANT-EXAMINER: Nguyen-Ba; Antony

ATTY-AGENT-FIRM: Glenn; Michael A. Wong; Kirk D.

ABSTRACT:

A method and system are provided for converting a computer program from a current version first language to a localized version in a target language. All resource

h e b b g e e e f e e ef b e

information of the program is stored in a resource dynamic link library (DLL). A current version resource DLL is separated from executable code and compared by a leverage tool to resource DLLs of a previous version of the computer program and to a previous target language translation. The leverage tool stores all resource data from the current version resource DLL as translation records in a resource database. The translation records may include translation instructions and comments. Strings in the current version resource DLL that were present in the previous version, and already translated in the previous target language resource DLL are stored in a new target language resource DLL. Strings which are not to be translated to the target language are locked. In the preferred embodiment, the new target language resource DLL and resource database are distributed to a localization vendor for translation. A translation tool allows a translator to translate the resource database to the target language. Locked strings are not translated and the already-translated strings are re-used.

13 Claims, 6 Drawing figures

☐ 14. Document ID: US 5963925 A

L11: Entry 14 of 27 File: USPT Oct 5, 1999

US-PAT-NO: 5963925
DOCUMENT-IDENTIFIER: US 5963925 A

TITLE: Electronic statement presentment system

DATE-ISSUED: October 5, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Kolling; Ray	Menlo Park	CA		
Occhino; Michael	Castro Valley	CA		
Roughgarden; Jeffrey D.	Redwood City	CA		
Hayward; James T.	Shawnigan Lake			CA

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Visa International Service Association	Foster City	CA			02

APPL-NO: 08/ 947629 [PALM]
DATE FILED: October 8, 1997

PARENT-CASE:
This application claims priority of provisional U.S. patent application Ser. No. 60/028,095, filed Oct. 9, 1996, entitled "Electronic Statement Presentation", by inventors Kolling, et al., which is hereby incorporated by reference in its entirety for all purposes.

INT-CL: [06] G06 F 17/60

h e b b g e e e f e e ef b e

US-CL-ISSUED: 705/40; 705/27, 705/34, 705/39, 705/43
US-CL-CURRENT: 705/40; 705/27, 705/34, 705/39, 705/43

FIELD-OF-SEARCH: 705/27, 705/34, 705/40, 705/30, 705/44, 705/39, 705/43, 379/91.01

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
3375500	March 1968	Tateisi et al.	705/43
3652795	March 1972	Wolf et al.	379/91.01
4186438	January 1980	Benson et al.	711/113
5007084	April 1991	Martena et al.	380/24
5025373	June 1991	Keyser et al.	380/24
5050207	September 1991	Hitchcock	379/93.19
5220501	June 1993	Lawlor et al.	380/24
5482185	January 1996	Chancey et al.	705/40
5699528	December 1997	Hogan	705/40
5715298	February 1998	Rogers	379/91.01
5774885	August 1998	Delfer	705/401
5832460	November 1998	Bednar et al.	705/27

OTHER PUBLICATIONS

"Online Fund System Among Main Banks", Oct., 1989, Money Circulation System.
"Money Circulation Data System Center", Electronic Banking, Oct. 31, 1986, pp. 35-41.
D.W. Davies and W.W. Price, translated by Tadahiro Ueno, "Electronic Money Movement and Intelligent Token", Dec. 5, 1985, Network Security, Chapter 10, pp. 251-263 and pp. 283-287.
"Notice of the Reason for Refusal", Mar. 31, 1998, Japanese Patent Office.
Mike Ter Maat, "The Economics of E-Cash", IEEE Spectrum 1997.
Howard Anderson, "Money and the Internet: A Strange New Relationship" IEEE Spectrum 1997.
"To Probe Further", IEEE Spectrum 1997.
"CyberCash's Secure Internet Payment Services", CyberCash, Inc., Reston, Virginia 22091.
Leslie Marable. "A Test Moves Net-Based Bill Payment a Step Closer", WebWeek, vol. Three, Feb. 3, 1997.
"Visa cPay: A Fully Electronic Solution for Biller Payment Delivery", Visa U.S.A. Inc. V4068-06-96, 1996.
Tekla S. Perry, "Electronic Money: Toward a Virtual Wallet", IEEE Spectrum, Feb. 1997.
Edward W. Kelley, Jr., "The Future of Electronic Money: A Regulator's Perspective", IEEE Spectrum, Feb. 1997.
Marvin A. Silbu, "Credits and Debits on the-Internet", IEEE Spectrum Feb. 1997.
David Chaum, et al., "Minting Electronic Cash", IEEE Spectrum Feb. 1997.
Peter S. Gemmell, "Traceable E-Cash", IEEE Spectrum Feb. 1997.
Stanley E. Morris, "Crime and Prevention: A Treasury Viewpoint", IEEE Spectrum Feb. 1997.
Baldwin et al., "Locking the E-Safe", IEEE Spectrum Feb. 1997.
Carol Hovengn Fancher, "In Your Pocket: Smartcards", IEEE Spectrum Feb. 1977,
Michael C. McChesney, "Banking in Cyberspace: An Investment in Itself", IEEE

Spectrum 1997.
Steven M.H. Wallmanm, "Technology Takes to Securities Trading", IEEE Spectrum 1997.

Alfred R. Berkeley, III, "Nasdaq's Technology Floor: Its President Takes Stock", IEEE Spectrum 1997.

ART-UNIT: 275

PRIMARY-EXAMINER: MacDonald; Allen R.

ASSISTANT-EXAMINER: Jeanty; Romain

ATTY-AGENT-FIRM: Beyer & Weaver, LLP

ABSTRACT:

An electronic statement presentment (ESP) system replaces the preparation and mailing of paper statements and invoices from a biller with electronic delivery. Electronic statements have the same look as paper statements as well as including video, audio, graphics, and custom enclosures. Statements are segmented into mandatory and optional components to minimize download time. The ESP system operates independently or is an enhancement to any suitable electronic bill payment system. A central switch computer coordinates template storage, validation, routing and message passing between billers, workstations and consumer financial institutions (CFI). A template authoring workstation (TAWS) creates a template of static biller information to serve as a basis for the electronic statement. The template is stored in a template library at the switch. The switch validates the template by sending it to a template validation workstation (TVAL). Batches of customer statement data are sent from a biller's legacy invoicing system to a statement origination workstation (SORG) along with a template identifier. The switch sends the template to the SORG where the customer data is validated by comparison to the template identified. The batch of customer statement data is sorted by a statement generation workstation (SGEN) identifier associated with each customer record. The sorted batches are sent to the switch where they are routed to the appropriate SGEN based upon the SGEN identifier. Each SGEN generates an electronic statement for each customer from the statement data and the appropriate template. A CFI associated with each SGEN delivers each electronic statement to the appropriate customer using a customer identifier in the statement data and uses any chosen medium.

46 Claims, 20 Drawing figures

☐ 15. Document ID: US 5790860 A

L11: Entry 15 of 27 File: USPT Aug 4, 1998

US-PAT-NO: 5790860
DOCUMENT-IDENTIFIER: US 5790860 A

TITLE: Method and apparatus for patching code residing on a read only memory device

DATE-ISSUED: August 4, 1998

INVENTOR-INFORMATION:

h e b b g e e e f e e ef b e

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wetmore; Russ	Santa Clara	CA		
Nguyen; Philip	Santa Cruz	CA		
Batista; Ricardo	Santa Clara	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Apple Computer, Inc.	Cupertino	CA			02

APPL-NO: 08/ 412293 [PALM]
DATE FILED: March 28, 1995

PARENT-CASE:
This is a divisional of application Ser. No. 08/058,877, filed May 6, 1993, now U.S. Pat. No. 5,481,713.

INT-CL: [06] G06 F 9/44

US-CL-ISSUED: 395/705
US-CL-CURRENT: 717/122

FIELD-OF-SEARCH: 395/700, 395/704, 395/705

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4542453	September 1985	Patrick et al.	
4610000	September 1986	Lee	365/189
4688173	August 1987	Mitarai et al.	364/405
4751703	June 1988	Picon et al.	371/10
4802119	January 1989	Heene et al.	364/900
4831517	May 1989	Crouse et al.	364/200
4982360	January 1991	Johnson et al.	364/900

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
15940	September 1992	WO	
00633	January 1993	WO	

OTHER PUBLICATIONS

IBM Technical Disclosure Bulletin, vol. 31, No. 1, Jun. 1988, pp. 294-298, "Dual Indirect RAM/ROM Jumptable for Firmware Update".
IBM Technical Disclosure Bulletin, vol. 34, No. 7, Dec., 1992, pp. 8-13, "Method and Mechanism for Dynamic Loader" See p. 11 line 41, p. 13 line 32.
Bradley et al., IBM Technical Disclosure Bulletin, vol. 27, No. 4A, Sep. 1984, pp. 2187-2188, "Method of Customizing Patches for Each Hardware Configuration".

h e b b g e e e f e e ef b e

No Author, IBM Technical Disclosure Bulletin, vol. 31, No. 1, June 1988, pp. 294-298, "Dual Indirect RAM/ROM Jumptables for Firmware Updates" see whole document.
No Author, IMB Technical Disclosure Bulletin, vol. 34, No. 7, Dec. 1992, pp. 8-13, "Method and Mechanism for Dynamic Loader" see p. 11 line41, p. 13 line 32.
Bradley et al. IBM Technical Disclosure Bulletin, vol. 27, No. 4A, Sep. 1984, pp. 2187-2188, "Method of Customizing Patches for Each Hardware Configuration" see whole document.

ART-UNIT: 236

PRIMARY-EXAMINER: Oberley; Alvin E.

ASSISTANT-EXAMINER: Park; Alice Y.

ATTY-AGENT-FIRM: Blakely, Sokoloff, Taylor & Zafman

ABSTRACT:

A method and apparatus for generating patching resources in an information processing system having operating instructions on a Read Only Memory Device. The present invention simplifies the patch generation and installation processes. A patch resource is generated and used by a patch installation process. Patch resources are generated for each ROM version by comparing previous ROM versions to the new ROM version. A patch resource is comprised of a plurality of entries, each of which defines a vector table address, an offset into the vector table and the routine to be inserted. By comparing routines between the ROM versions, routines which are different or new are identified. These routines will become patch resource entries. For patch installation, the ROM version number for the installed ROM is determined; the proper patching resource is retrieved, and the patch resource entries cause the patches to be installed. Patch installation is performed by the steps of modifying vector tables to include the addresses for the new routines.

3 Claims, 8 Drawing figures

☐ 16. Document ID: US 5724272 A

L11: Entry 16 of 27 File: USPT Mar 3, 1998

US-PAT-NO: 5724272
DOCUMENT-IDENTIFIER: US 5724272 A

TITLE: Method and apparatus for controlling an instrumentation system

DATE-ISSUED: March 3, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Mitchell; Bob	Travis County	TX		
Andrade; Hugo	Travis County	TX		
Pathak; Jogen	Travis County	TX		
DeKey; Samson	Travis County	TX		

h e b b g e e e f e e ef b e

Shah; Abhay	Travis County	TX
Brower; Todd	Travis County	TX

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
National Instruments Corporation	Austin	TX			02

APPL-NO: 08/ 238480 [PALM]
DATE FILED: May 4, 1994

INT-CL: [06] G06 F 19/00

US-CL-ISSUED: 364/579; 395/674
US-CL-CURRENT: 702/123; 718/104

FIELD-OF-SEARCH: 364/579, 364/580, 364/191, 395/62, 395/500, 395/700, 395/970, 395/674, 395/681-685

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5337262	August 1994	Luthi et al.	364/580
5359546	October 1994	Hayes et al.	364/579
5361336	November 1994	Atchison	
5390325	February 1995	Miller	395/700

OTHER PUBLICATIONS

Hewlett Packard SICL Standard Instrument Control Library for C Programming, Lee Atchison, VXI Systems Division, Hewlett Packard Company, Jan. 21, 1994, Revision 3.9, pp. i-iii, 1-136, Index.
Marketing brochures and product literature for instrumentation systems produced by Bruel & Kjaer, including the Modular Test System Type 3538, available at the Auto TestCon instrumentation conference in 1991.
Slide presentation on the Bruel & Kjaer Modular Test system.

ART-UNIT: 244

PRIMARY-EXAMINER: Trammell; James P.

ATTY-AGENT-FIRM: Conley, Rose & Tayon Hood; Jeffrey C.

ABSTRACT:

A method and apparatus for controlling instrumentation systems and for providing a user with the capability to develop instrument drivers and application software for controlling instrumentation systems. The present invention provides a system including a software architecture which defines the control and management of an instrumentation system. The method of the present invention utilizes a device resource independence approach whereby the individual capabilities of devices are broken down into a plurality of objects called resources and these resources are then used to develop instrument drivers or instrument control applications. The

h e b b g e e e f e e ef b e

method of the present invention also uses object oriented technology which allows device resources to be easily combined to create higher level applications. The present invention is independent of I/O interface type, operating system, and programming language while also providing a common look and feel and consistent API to the user. The present invention includes novel methods for access control, event handling, resource management, and resource addressing, among others.

177 Claims, 44 Drawing figures

☐ 17. Document ID: US 5701480 A

L11: Entry 17 of 27 File: USPT Dec 23, 1997

US-PAT-NO: 5701480
DOCUMENT-IDENTIFIER: US 5701480 A

TITLE: Distributed multi-version commitment ordering protocols for guaranteeing serializability during transaction processing

DATE-ISSUED: December 23, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Raz; Yoav	Newton	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Digital Equipment Corporation	Maynard	MA			02

APPL-NO: 08/ 047271 [PALM]
DATE FILED: April 14, 1993

PARENT-CASE:
RELATED APPLICATIONS The present application is a continuation-in-part of Yov Raz, U.S. patent application Ser. No. 07/778,254 filed Oct. 17, 1991, now abandoned entitled "Guaranteeing Global Serializability By Applying Commitment Ordering Selectively to Global Transactions." The prior application Ser. No. 07/778,254 shows that the component operations of a number of global transactions can be distributed and scheduled for execution using any kind of resource manager with a local scheduler that ensures local serializability, yet global consistency can be maintained by enforcing a global transaction commitment ordering that is consistent with the order of conflicts among global transactions, including indirect conflicts caused by local transactions. Conformance to such a global transaction commitment ordering in such a distributed transaction processing system guarantees the serializability of the combined (global) schedule. The present application includes disclosure from Yoav Raz, U.S. patent application Ser. No. 07/703,394 filed May 21, 1991, now abandoned entitled "Commitment ordering for Guaranteeing Serializability Across Distributed Transactions," which shows that if global atomicity of transactions is achieved via an atomic commitment protocol, then a "commitment ordering" property of transaction histories is a sufficient condition for global serializability. The performance of conflicting operations is scheduled according to available resources, and the "commitment ordering" property is enforced by ensuring that the results of the conflicting operations are committed in the same

order as the order of performance of the conflicting operations. The present application also includes disclosure from Spiro et al., U.S. patent application Ser. No. 07/717,212 filed Jun. 18, 1991, now U.S. Pat. No. 5,369,757 entitled "Recovery Logging in the Presence of Snapshot Files by Ordering of Buffer Pool Flushing," which discloses aspects of a multi-version transaction processing system that permits read-only transactions to bypass write locks in the "Rdb/VMS" (Trademark) brand of operating system sold by Digital Equipment Corporation of Maynard, Mass. 01754-1499.

INT-CL: [06] G06 F 15/00

US-CL-ISSUED: 395/671; 395/182.17
US-CL-CURRENT: 718/101; 714/19

FIELD-OF-SEARCH: 395/700, 395/650, 395/600, 395/610, 395/671, 395/182.17

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4224664	September 1980	Trinchieri	364/200
4249241	February 1981	Aberle et al.	364/200
4627019	December 1986	Ng	364/900
4881166	November 1989	Thompson et al.	364/200
5193188	March 1993	Franaszek et al.	395/650
5212788	May 1993	Lomet et al.	395/600
5263156	November 1993	Bowen et al.	395/600
5369757	November 1994	Spiro et al.	395/575
5504900	April 1996	Raz	395/650

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0457112	November 1991	EP	

OTHER PUBLICATIONS

H.F. Korth & A. Silberschatz, Database System Concepts, McGraw-Hill, Inc., United States, pp. xiv-xix, 490-503 (1991).
Vassos Hadzilacos, "A Knowledge Theoretic Analysis of Atomic Commitment Protocols," Proc. of the Sixth ACM Symposium on Principles of Database Systems, Association for Computing Machinery, New York, NY, Mar. 23-25, 1987. pp. 129-134.
Joseph Y. Halpern, "Using Reasoning about Knowledge to Analyze Distributed Systems," Research Report RJ 5522 (56421) Mar. 3, 1987, Computer Science, IBM Almaden Research Center, San Jose, California, 1987.
Lampson et al., "Crash Recovery in a Distributed Data Storage System," Technical Report, Xerox, Palo Alto Research Center, Palo Alto, California, 1986.
Litwin et al., "Flexible Concurrency Control Using Value Dates," in Integration of Information Systems: Bridging Heterogeneous Databases, ed. A Gupta, IEEE Press. IEEE, New York, NY, 1989, pp. 144-145.
Weihl, "Distributed Version Management for Read-Only Actions," IEEE Transactions on

Software Engineering, vol. SE-13, No. 1, IEEE, New York, NY, Jan. 1987, pp. 55-64.
Christos Papadimitriou, The Theory Database Concurrency Control, Computer Science Press, Inc., Rockville, Maryland, (1986), pp. 93-158, 201-224.
Vassos Hadzilacos, "A Theory of Reliability in Database Systems, " Journal of the ACM, vol. 35, No. 1, Jan., 1988, pp. 121-145, Association for Computing Machinery, New York, New York.
Agrawal et al., "Modular Synchronization in Multiversion Databases: Version Control and Concurrency Control," Proc. of the 1989 ACM SIGMOD Int. Conf. on Management of Data, pp. 408-417, Portland, Oregon, Jun., 1989 Association for Computing Machinery, New York N.Y.
Raghavan et al., "Database Availability for Transaction Processing," Digital Technical Journal, vol. 3, No. 1, Winter 1991, pp. 65-69, Digital Equipment Corp., Maynard, Mass. 1991.
Bober et al., "Multiversion Query Locking," Proc. of the Eighteenth Int. Conf. on Very Large Databases, Vancouver, British Columbia, (Aug. 1992), pp. 497-510.
Agrawal et al., "An Approach to Eliminate Transaction Blocking in Locking Protocols," Proc. of the Eleventh ACM Symposium on Principles of Database Systems, San Diego Calif. (Jun. 1992), pp. 223-235.
Bober et al., "On Mixing Queries and Transactions Via Multiversion Locking, " Proc. of the Eighth Int. Conf. on Data Engineering, Tempe, (Feb. 1992), pp. 535-545.
Vaijalainen et al., "Prepare and Commit Certification for Decentralized Transaction Management in Rigorous Heterogeneous Multidatabase," IEEE, New York, N.Y. (1992), pp. 470-479.
Mohan et al., "Efficient and Flexible Methods for Transient Versioning of Records to Avoid Locking by Read-Only Transactions," 1992 ACM SIGMOND Conf. Assoc.-Computing Machinery, New York, NY (Jun. 1992), pp. 124-133.
Silberschatz et al., "Database Systems: Achievements and Opportunities," Communications of the ACM, vol. 34, No. 10, Association for Computing Machinery, New York, N.Y., Oct., 1991, pp. 110, 120.
Breitbart et al., "On Rigorous Transaction Scheduling," IEEE Transactions on Software Engineering, vol. 17, No. 9, IEEE, New York, NY (Sept. 1991).
Agrawal et al., "Performance Characteristics of Protocols With Ordered Shared Locks," Proceedings of the Seventh IEEE International Conference on Data Engineering, Institute of Electrical and Electronics Engineers, New York, N.Y. (Apr. 1991), pp. 592-601.
Georgakopoulos et al., "On Serializability of Multidatabase Transactions through Forced Local Conflicts," Proceedings of the Seventh Int. Conf. on Data Engineering, Kobe, Japan (Apr. 1991).
L. Hobbs et al., "Rdb/VMS--A Comprehensive Guide", Digital Press, Digital Equipment Corporation, Maynard, Mass., 1991.
David Lomet, "Consistent Timestamping for Transactions in Distributed Systems," TR CRL 90/3, Digital Equipment Corporation, CRL, Cambridge, Mass. (Sep. 1990).
Sheth et al., "Federated Database System for Managing Distributed Hetergeneous and Autonomous Databases," ACM Computing Surveys, Col. 22, No. 3, Assoc. for Computing Machinery, New York, N.Y. (Sep. 1990), pp. 183-236.
Y. Breitbart, "Multidatabase Inoperability," Sigmond Record, vol. 19, No. 3, Assoc. for Computing Machinery, New York, N.Y. (Sep. 1990), pp. 53-60.
Breitbart et al., "Reliable Transaction Management in a Multidatabase System," Proc. of the ACM SIGMOD Conf. on Management of data, Atlantic City, New Jersey (Jun. 1990), pp. 215-224.
Agrawal et al., "Locks with Constrained Sharing," Proceedings of the Nineth ACM Symposium on Principles of Database Systems, Association for Computing Machinery, New York, N.Y. (Apr. 1990), pp. 85-93.
Elmagarmid et al., "A Paradigm for Concurrency Control in Heterogeneous Distributed Database Systems," IEEE, New York, N.Y. (1990), pp. 17-46.
Georgakopoulos et al., "Transaction Management in Multidatabase Systems," Technical Report #UH-CS-89-20, Depart. of Computer Science, University of Houston, Houston, Texas (Sep. 1989).
Calton Pu, "Transactions across Heterogeneous Databases: the Superdatabase Architecture," Department of Computer Science, Columbia University, New York, N.Y.

h e b b g e e e f e e ef b e

(Jun. 1988), pp. 1-18.
Carlton Pu, "Superdatabase for Composition of Heterogeneous Databases," 4th International Conference on Data Engineering, Los Angeles, California IEEE, New York, N.Y. (Feb. 1-5, 1988), pp. 548-555.
Garcia-Molina et al., "Node Autonomy in Distributed Systems," IEEE, New York, N.Y. (1988), pp. 158-166.
William E. Weihl, "Distributed Version Management for Read-Only Actions," IEEE Transactions on Software Engineering, vol. SE-13, No. 1, IEEE, New York, N.Y. (Jan. 1987), pp. 55-64.
Bernstein et al., "Concurrency Control and Recovery in Database Systems," Addison-Wesley, Reading, Mass. (1987), pp. 58-78.
Chan et al., "Implementing Distributed Read-Only Transactions," IEEE Transactions on Software Engineering, vol. SE-11, No. 2, IEEE, New York, N.Y. (Feb. 1985), pp. 205-212.
J. Eliot B. Moss, Nested Transactions, The MIT Press, Cambridge, Mass. (1985).
Kung et al., "On Optimistic Methods for Concurrency Control," ACM Transactions on Database Systems, vol. 6, No. 2, Columbus, Ohio (Jun. 1981), pp. 213-226.
Jim Gray, "Operating Systems: An Advanced Course," Lecture Notes in Computer Science 60, IBM Research Laboratory, San Jose Calif. (1978), pp. 393-481.
OS/Texas-11 Programming Manual, Digital Equipment Corporation, Maynard, Mass. (Apr. 20, 1972).
Roger Bamford, "Using Multiversioning to Improve Performance Without Loss of Consistency," Abstract, Proc. of the ACM SIGMOND Int. Conf. on Management of Data, San Diego, Calif. (Jun. 1992), p. 164.
Yoav Raz, "The Principle of Commitment Ordering, or Guaranteeing Serializability in a Heterogeneous Environment of Multiple Autonomous Resource Managers, Using Atomic Commitment", in the Proc. of the Eighteenth Int. Conf. on Very Large Data Bases, pp. 292-312, Vancouver, Canada, Aug. 1992.
H. Garcia-Molina, G. Wiederhold, "Read-Only Transactions in a Distributed Database", ACM Trans. Database Systems 7(2), Association for Computing Machinery, New York, N.Y. Jun. 1982, pp. 209-234.
J. Puustjarvi, "Distributed Management of Transactions in Heterogeneous Distributed Database Systems", BIT, 31(2), pp. 406-420, 1991.
Agrawal et al., "The Performance of Protocols based on Locks with Ordered Sharing," University of California, Department of Computer Science TRCS 92-9, Santa Barbara, California, Apr., 1992.
Breitbart et al., "Overview of Multi-Database Transaction Management," University of Kentucky, Lexington, Kentucky.
Breitbart et al., "Complexity of Global Transaction Management in Multidatabase Systems," Technical Report No. 198-91, University of Kentucky, Lexington, Kentucky, Nov. 1, 1991.
Breitbart et al., "Using Delayed Commitment in Locking Protocols for Real-Time Databases," Proc. of the ACM SIGMOD Int. Conf. on Management of of Data, San Diego, Calif. (Jun. 1992), pp. 104-112.
Breitbart et al., "Strong Recoverabilty in Multidatabase Systems," Proceedings, 2nd International Workshop on Research Issues in Database Engineering (RIDE-TQP), Phoenix, AZ, February 1992.

ART-UNIT: 236

PRIMARY-EXAMINER: Kriess; Kevin A.

ASSISTANT-EXAMINER: Chaki; Kakali

ABSTRACT:

In a multi-version database, copies of prior committed versions (snapshots) are kept for access by the read-only transactions. The read-write transactions are selectively aborted to enforce an order of commitment of read-write transactions

h e b b g e e e f e e ef b e

that is the same as an order of conflicts among the read-write transactions. In a preferred embodiment, the read-write transactions are serialized by maintaining and referencing a graph of conflicts among read-write transactions, and the read-only transactions are serialized by a timestamp mechanism for selection of the snapshots to be read. Each time that a read-write transaction is committed, the read-write transaction is assigned a unique timestamp that is used to timestamp all resources committed by the read-write transaction. Upon starting, each read-only transaction is also assigned a timestamp. Each read-only transaction reads only the latest committed versions of all resources, that are timestamped earlier than the timestamp of the read-only transaction. In a multiprocessing system, the timestamps are issued to global coordinators and distributed locally with atomic commit messages and global queries. Moreover, read-write transactions may selectively access a hierarchy of uncommitted versions to prepare for various possible commitment orders. The hierarchy defines a path for record access and for cascading aborts. A plurality of mutually-conflicting uncommitted versions may be prepared for each transaction to prepare for all possible commitment orders.

34 Claims, 51 Drawing figures

☐ 18. Document ID: US 5678039 A

L11: Entry 18 of 27 File: USPT Oct 14, 1997

US-PAT-NO: 5678039
DOCUMENT-IDENTIFIER: US 5678039 A

TITLE: System and methods for translating software into localized versions

DATE-ISSUED: October 14, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hinks; Paul	Santa Cruz	CA		
Lok; James Shian Hwa	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Borland International, Inc.	Scotts Valley	CA			02

APPL-NO: 08/ 316690 [PALM]
DATE FILED: September 30, 1994

INT-CL: [06] G06 F 17/30, G06 F 17/28

US-CL-ISSUED: 395/604; 395/603, 395/606, 395/619, 395/333, 395/335, 395/758, 395/795
US-CL-CURRENT: 707/4; 345/835, 707/203, 707/3, 707/6, 715/533

FIELD-OF-SEARCH: 395/600, 395/153, 395/603, 395/604, 395/606, 395/619, 395/333, 395/335, 395/758, 395/795, 364/200, 364/419.05, 364/900

PRIOR-ART-DISCLOSED:

h e b b g e e e f e e ef b e

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
3611316	October 1971	Woodrum	340/172.5
4456969	June 1984	Herzik et al.	364/900
4566078	January 1986	Crabtree	364/900
4575798	March 1986	Lindstrom et al.	364/300
4731735	March 1988	Borgendale et al.	364/200
4809158	February 1989	McCauley	364/200
4870402	September 1989	DeLuca et al.	340/825
5072386	December 1991	Gerneau et al.	364/419
5148541	September 1992	Lee et al.	395/600
5175803	December 1992	Yeh	395/100
5181162	January 1993	Smith et al.	364/419
5243519	September 1993	Andrews et al.	364/419.05
5274805	December 1993	Ferguson et al.	395/600
5416903	May 1995	Malcolm	395/155
5490061	February 1996	Tulin et al.	364/419.02

OTHER PUBLICATIONS

Petzold, C., Programming Windows, Second Edition, 1990, pp. 125-131 (Chapter 3: The Keyboard).
William S. Hall, "Adapt Your Program for Worldwide Use With Windows International Support", Microsoft Systems Journal, Nov./Dec. 1992, pp. 29-58.

ART-UNIT: 237

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Lewis; C.

ATTY-AGENT-FIRM: Smart; John A. Ritter; Michael J.

ABSTRACT:

A Software Translation Kit (STK) system having a shell, TShell, coupled to an Export/Import module and various Editors is described. The Export/Import module itself includes a parsing engine to extract strings and translatable information from application programs. It functions as a front end parser to "translatable" sources, providing data conversion as needed. The STK system provides a standard interface and set of tools which can be used to localize graphic user interface products. By employing a datacentric approach, the system provides a standard platform which allows translators to act independently of the product they are translating.

43 Claims, 29 Drawing figures

h e b b g e e e f e e ef b e

☐ 19. Document ID: US 5603034 A

L11: Entry 19 of 27 File: USPT Feb 11, 1997

US-PAT-NO: 5603034
DOCUMENT-IDENTIFIER: US 5603034 A

TITLE: Graphical resource editor for software customization

DATE-ISSUED: February 11, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Swanson; Sara J.	Mountain View	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 479262 [PALM]
DATE FILED: June 7, 1995

PARENT-CASE:
This application is a continuation of application Ser. No. 07/941,579, filed Sep. 8, 1992, now abandoned.

INT-CL: [06] G06 F 3/14

US-CL-ISSUED: 395/701; 395/333
US-CL-CURRENT: 717/111; 345/781, 345/866, 717/113

FIELD-OF-SEARCH: 395/155, 395/DIG.1, 395/DIG.2, 395/700

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4648046	March 1987	Copenhaver et al.	364/518
5097411	March 1992	Doyle et al.	395/600
5115501	May 1992	Kerr	395/600
5327529	July 1994	Fults et al.	395/155
5369778	November 1994	San Soucie et al.	395/800

OTHER PUBLICATIONS

Computer file printout re: "Motif Resource Editor".
Computer file printout re: Editres(1) from UNIX Programmer's Manual, X Version 11, Release 5.
"Using the X-toolkit or How to Write a Widget", McCormack et al., Summer USENIX '88 pp. 1-13.

h e b b g e e e f e e ef b e

"Intrinsics of the X toolkit", T. Lainhart, Dr. Dobb's Journal, Feb. 1991, v16 n2 p. 94(12).
"Skin and Bones", Eric Giguere, Computer Language, Apr. 1990 v7 n4 p. 43(11).
"Simplicity and Productivity", Paul Aseute, Unix Review, vol. 6 n6, Sep. 1988, p. 56(6).
"Making Smalltalk with widgets"; K. E. Ayers, Dr. Dobb's Journal, May 1991, v16 n5 p. 64(9).
"Using the X-toolkit", J. McCormack et al., Summer Usenix 88, pp. 1-13.

ART-UNIT: 236

PRIMARY-EXAMINER: Kriess; Kevin A.

ASSISTANT-EXAMINER: Chaki; Kakali

ATTY-AGENT-FIRM: Baker, Maxham, Jester & Meador

ABSTRACT:

A graphical resource editor for selectively modifying graphical resources in software applications provides a main window graphical user interface object for interaction with the graphical resource editor. The main window contains a resource category selection object including a list of selectable resource category objects. These objects contain resource category descriptors corresponding to categories of editable resources. The resource category selection object provides a user activatable interface for selecting among the list of resource category objects an editable resource category. The graphical resource editor further includes a system responsive to user activation of the resource category selection object for generating in the main window a list of resource descriptors corresponding to selected category of editable resources. A plurality of resource value display fields are provided in the main window for displaying resource values representing the status of the selected category of editable resources. Also provided in the main window is a set of resource value selection objects providing user activatable interfaces for setting editable resource values. Customization of software applications may performed statically by saving resource edits to an application resource file, or dynamically by applying resource edits on-the-fly to an application running concurrently with the graphical resource editor.

3 Claims, 18 Drawing figures

☐ 20. Document ID: US 5600778 A

L11: Entry 20 of 27 File: USPT Feb 4, 1997

US-PAT-NO: 5600778
DOCUMENT-IDENTIFIER: US 5600778 A
**** See image for Certificate of Correction ****

TITLE: Graphical resource editor for software customization

DATE-ISSUED: February 4, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY

h e b b g e e e f e e ef b e

Swanson; Sara J.	Mountain View	CA
Andrews; Robert F.	San Jose	CA
Mandelstam; Gary M.	Cupertino	CA

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 525895 [PALM]
DATE FILED: September 8, 1995

PARENT-CASE:
This application is a file wrapper continuation, of application Ser. No. 07/942,204, filed Sep. 8, 1992, now abandoned.

INT-CL: [06] G06 F 3/14

US-CL-ISSUED: 395/333; 395/340, 395/352, 395/356, 395/335
US-CL-CURRENT: 345/762; 345/781, 345/810, 345/853

FIELD-OF-SEARCH: 395/155, 395/157, 395/160, 395/159, 395/161, 395/156, 345/119, 345/146, 345/902

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4866638	September 1989	Cosentino et al.	395/159
5041992	August 1991	Cunningham et al.	395/155 X
5202828	April 1993	Vertelney et al.	395/159 X
5249263	September 1993	Yanker	395/155 X
5297250	March 1994	Leroy et al.	395/157
5335320	August 1994	Iwata et al.	395/155
5347627	September 1994	Hoffmann et al.	395/157
5347629	September 1994	Barrett et al.	395/161
5371844	December 1994	Andrew et al.	395/155
5388202	February 1995	Squires et al.	395/157
5398312	March 1995	Hoffmann	395/156
5430836	July 1995	Wolf et al.	395/155
5522024	May 1996	Hiraga et al.	395/155

OTHER PUBLICATIONS

Microsoft Windows User's Guide, Version 3.0, Microsoft Corporation, 1990, Chapter 5 Control Panel, pp. 147-158, 177-180.

ART-UNIT: 245

PRIMARY-EXAMINER: Bayerl; Raymond J.

h e b b g e e e f e e ef b e

ATTY-AGENT-FIRM: Baker, Maxham, Jester & Meador

ABSTRACT:

A graphical resource editor for selectively modifying graphical resources in software applications provides a main window graphical user interface object for interaction with the graphical resource editor. The main window contains a resource category selection object including a list of selectable resource category objects. These objects contain resource category descriptors corresponding to categories of editable resources. The resource category selection object provides a user activatable interface for selecting among the list of resource category objects an editable resource category. The graphical resource editor further includes a system responsive to user activation of the resource category selection object for generating in the main window a list of resource descriptors corresponding to the selected category of editable resources. A plurality of resource value display fields are provided in the main window for displaying resource values representing the status of the selected category of editable resources. Also provided in the main window is a set of resource value selection objects providing user activatable interfaces for setting editable resource values. Customization of software applications may performed statically by saving resource edits to an application resource file, or dynamically by applying resource edits on-the-fly to an application running concurrently with the graphical resource editor.

21 Claims, 18 Drawing figures

☐ 21. Document ID: US 5572672 A

L11: Entry 21 of 27 File: USPT Nov 5, 1996

US-PAT-NO: 5572672
DOCUMENT-IDENTIFIER: US 5572672 A

TITLE: Method and apparatus for monitoring data processing system resources in real-time

DATE-ISSUED: November 5, 1996

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Dewitt; Jimmie E.	Georgetown	TX		
Holck; Timothy M.	Austin	TX		
Summers; James H.	Round Rock	TX		
Emrick; Samuel L.	Austin	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 500656 [PALM]
DATE FILED: July 12, 1995

h e b b g e e e f e e ef b e

PARENT-CASE:
This is a continuation of application Ser. No. 08/263,153 filed on Jun. 20, 1994, U.S. Pat. No. 5,463,775, which is a continuation of application Ser. No. 07/713,484 filed on Jun. 10, 1991, now abandoned. A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

INT-CL: [06] G06 F 12/00, G06 F 13/00

US-CL-ISSUED: 395/184.01; 395/431, 395/494, 395/497.02, 395/427, 395/821, 395/616, 364/550, 364/569, 364/DIG.1, 364/238.4, 364/242.4, 364/251.3, 364/252
US-CL-CURRENT: 714/47; 702/176, 707/200, 710/1, 711/100, 711/104, 711/167, 711/171

FIELD-OF-SEARCH: 395/184.01, 395/431, 395/494, 395/497.02, 395/427, 395/821, 395/600, 364/550, 364/569, 364/DIG.1

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4019040	April 1977	Thompson	235/92
4291376	September 1981	McCahill	364/483
4296727	October 1981	Bryan	126/116
4373184	February 1983	Lambregts	364/434
4459656	July 1984	Wilder, Jr.	364/200
4485440	November 1984	Duff et al.	364/300
4499539	February 1985	Vosacek	395/600
4533910	August 1985	Sukonick et al.	340/721
4533997	August 1985	Furgerson	364/200
4590550	May 1986	Eilert et al.	364/200
4660130	April 1987	Bartley et al.	395/419
4701848	October 1987	Clyde	
4821178	April 1989	Levin et al.	364/200
4823290	April 1989	Fasack et al.	364/550
4878183	October 1989	Ewart	364/227
4905171	February 1990	Kiel et al.	364/551.01
4937743	June 1990	Rassman et al.	364/401
5081577	January 1992	Hatle	364/200
5153837	October 1992	Shaffer et al.	395/325
5167030	November 1992	Spilo	395/700
5214772	May 1993	Weinberger et al.	395/575
5237681	August 1993	Kagan et al.	395/600
5241672	August 1993	Slomcenski et al.	395/600
5265252	November 1993	Rawson, III et al.	395/700
5359713	October 1994	Moran et al.	395/200
5463775	October 1995	Dewitt et al.	395/184.01

h e b b g e e e f e e ef b e

OTHER PUBLICATIONS

"General Purporse Data Collection Method", IBM TDB, vol. 16, No. 6, 11-73, pp. 1796-1798.
"Structure of Performance Monitor for Distributed Computer Systems", IBM TDB, vol. 20, No. 11B, Apr. 1978, pp. 5060-5065.
"Computer with Intergral Function Monitor", IBM TDB, vol. 10, No. 11, Apr. 1968, pp. 1700-1703.
"Performance Evaluator for Operating System", IBM TDB, vol. 16, No. 1, Jun. 1973, pp. 110-118.
"General Trace Facility", IBM TDB, vol. 15, No. 8, Jan. 1973, pp. 2446-2448.
"Definition and Measurement Method of `Working Set` When Analyzing Memory Utilization in OS/2," IBM TDB, vol. 33, No. 2, Jul. 1990, p. 186.
"Software Tool for Reducing Page Fault Frequency", IBM TDB, vol. 32, No. 5B, Oct. 1989, pp. 464-466.
"Performance Monitor of Small Remote Computer Systems", IBM TDB, vol. 19, No. 6, Nov. 1976, pp. 2386-2388.
"Memory Usage Estimator", IBM TDB, vol. 16, No. 1, Jun. 1973, pp. 284-285.
"Functional Working Sets", IBM TDB, vol. 19, No. 4, Sep. 1976, pp. 1363-1364.
"Working Set Determination", vol. 26, No. 9, Feb. 1984, p. 4761.
"Working-Set Coprocessor", IBM TDB, vol. 32, No. 4A, Sep. 1989, pp. 240-242.
"Memory Utilization Monitor", IBM TDB, vol. 15, No. 5, Oct. 1972, pp. 1709-1712.
"The OSRM2 System", OS/2 Resource Monitor System, Version 1.1, Copyright 1989-1990, C.O.L. Consulting Ltd.
"Monitoring Distributed Embedded Systems" by Ray Ford, 1990 pp. 237-244.

ART-UNIT: 237

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Homere; Jean R.

ATTY-AGENT-FIRM: Bailey; Wayne P. Henkler; Richard A.

ABSTRACT:

A graphical system resource monitor is provided to depict, in real-time, a data processing system's internal resource utilization. A window or viewport of a data processing system displays user specified internal system resources, such as memory, CPU, or peripheral device availability/utilization. This graphical representation of the `state` of the data processing system's resources is maintained in real-time, while the impact on the system's performance in providing such information is kept to a minimum. This is accomplished through a combination of various techniques, including specialized device drivers for the respective devices coupled with a unique data reduction technique. The graphical results of these resource monitors are continually updated in real-time. This real-time support provides an immediate and accurate representation of the internal operations of the data processing system. Further, these resources can be monitored at the process level of a multiprocessing system. These representations can be used by a user to identify, isolate, and fine-tune the data processing system's resources to improve the overall efficiency of the system being monitored.

11 Claims, 18 Drawing figures

h e b b g e e e f e e ef b e

☐ 22. Document ID: US 5546586 A

L11: Entry 22 of 27 File: USPT Aug 13, 1996

US-PAT-NO: 5546586
DOCUMENT-IDENTIFIER: US 5546586 A

TITLE: Method and apparatus for vectorizing the contents of a read only memory device without modifying underlying source code

DATE-ISSUED: August 13, 1996

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wetmore; Russ	Santa Clara	CA		
Nguyen; Philip	Santa Cruz	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Apple Computer, Inc.	Cupertino	CA			02

APPL-NO: 08/ 058876 [PALM]
DATE FILED: May 6, 1993

INT-CL: [06] G06 F 9/44

US-CL-ISSUED: 395/700; 364/280, 364/280.4, 364/281.3, 364/DIG.1
US-CL-CURRENT: 717/122

FIELD-OF-SEARCH: 395/700, 395/650, 364/280, 364/280.1, 364/280.4, 364/280.5, 364/281.3

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4542453	September 1985	Patrick et al.	395/375
4610000	September 1986	Lee	365/189
4688173	August 1987	Mitarai et al.	364/405
4751703	June 1988	Picon et al.	371/10
4802119	January 1989	Heene et al.	364/900
4831517	May 1989	Crouse et al.	364/200
4982360	January 1991	Johnson et al.	364/900
5193180	March 1993	Hastings	395/575
5297291	March 1994	Murphy	395/700

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL

h e b b g e e e f e e ef b e

15940	September 1992	WO
00633	January 1993	WO

OTHER PUBLICATIONS

No Author, IBM Technical Disclosure Bulletin, vol. 31, No. 1, Jun. 1988, pp. 294-298, "Dual Indirect RAM/ROM Jumptables for Firmware Updates" see whole document.
No Author, IBM Technical Disclosure Bulletin, vol. 35, No. 7, Dec. 1992, pp. 8-13, "Method and Mechanism for Dynamic Loader" see p. 11 line 41, p. 13 line 32.
Bradley et al. IBM Technical Disclosure Bulletin, vol. 27, No. 4A, Sep. 1984, pp. 2187-2188, "Method of Customizing Patches for Each Hardware Configuration" see whole document.
IBM Technical Disclosure Bulletin, vol. 31, No. 1, "Dual Indirect RAM/ROM Jump Tables for Firmware Updates", Jun. 1988, pp. 294-298.
Jourdain, "Programmer's Problem Solver for the IBM PC, XT & AT", 1986, pp. 21-25.

ART-UNIT: 236

PRIMARY-EXAMINER: Kriess; Kevin A.

ASSISTANT-EXAMINER: Butler; Dennis M.

ATTY-AGENT-FIRM: Blakely, Sokoloff, Taylor & Zafman

ABSTRACT:

A method and apparatus for generating an object file that facilitates patching and the introduction of new function. The present invention accomplishes this without disturbing the original source file. The present invention is particularly useful in the generation of programs that will exist on a static device such as a Read Only Memory (ROM) device. The present invention requires that access to routines in the object file be referenced through a vector table located in Random Access Memory (RAM). If a routine in ROM must be patched (i.e. replaced) or if new function is added, the vector table is modified. Modification may be either changing the contents of an existing entry (replacement) or adding a new entry (new function). Generally, this modification involves the steps of: identifying the entry points in the object file to create a vector source table, generating a vector object table from the vector source table; generating a symbol table from the vector object table; comparing entry points in the object files to entries in the symbol table; when a match is found, modifying the entry point of the object file to reference a corresponding entry in the vector table. Since the only the object file is modified, the original source file is not disturbed.

4 Claims, 8 Drawing figures

☐ 23. Document ID: US 5537596 A

L11: Entry 23 of 27 File: USPT Jul 16, 1996

US-PAT-NO: 5537596
DOCUMENT-IDENTIFIER: US 5537596 A

TITLE: Method and apparatus for overriding resource maps in a computer system

h e b b g e e e f e e ef b e

DATE-ISSUED: July 16, 1996

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Yu; Dean T.	Cupertino	CA		
Derossi; Christopher S.	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Apple Computer, Inc.	Cupertino	CA			02

APPL-NO: 08/ 425251 [PALM]
DATE FILED: April 17, 1995

PARENT-CASE:
RELATED APPLICATIONS This is a continuation of co-pending application Ser. No. 08/019,600 filed on Feb. 19, 1993.

INT-CL: [06] G06 F 9/06

US-CL-ISSUED: 395/700; 364/251.5, 364/254.2, 364/222.8, 364/DIG.1
US-CL-CURRENT: 717/168; 713/1, 713/100

FIELD-OF-SEARCH: 395/700, 364/300, 364/200, 364/DIG.1

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4604690	August 1986	Crabtree et al.	364/200
4623963	November 1986	Phillips	364/200
5210875	May 1993	Bealkowski et al.	395/700
5247683	September 1993	Holmes et al.	395/700
5257368	October 1993	Benson et al.	395/600
5257387	October 1993	Richek et al.	395/800
5261080	November 1993	Khoyi et al.	395/500
5311424	May 1994	Mukherjee et al.	364/401
5325532	June 1994	Crosswy et al.	395/700
5327553	July 1994	Jewett et al.	395/575
5355489	October 1994	Bealkowski et al.	395/700

ART-UNIT: 236

PRIMARY-EXAMINER: Oberley; Alvin E.

ASSISTANT-EXAMINER: Banankhah; Majid A.

ATTY-AGENT-FIRM: Carr, DeFilippo & Ferrell

h e b b g e e e f e e ef b e

ABSTRACT:

An improved method and apparatus for defining resources in a computer system is presented whereby resource maps in a computer system can be selectively updated by adding resources and superseding resources in an existing resource map by providing a new resource map which overrides the prior resource map.

12 Claims, 6 Drawing figures

☐ 24. Document ID: US 5481713 A

L11: Entry 24 of 27 File: USPT Jan 2, 1996

US-PAT-NO: 5481713
DOCUMENT-IDENTIFIER: US 5481713 A

TITLE: Method and apparatus for patching code residing on a read only memory device

DATE-ISSUED: January 2, 1996

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wetmore; Russ	Santa Clara	CA		
Nguyen; Philip	Santa Cruz	CA		
Batista; Ricardo	Santa Clara	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Apple Computer, Inc.	Cupertino	CA			02

APPL-NO: 08/ 058877 [PALM]
DATE FILED: May 6, 1993

INT-CL: [06] G06 F 9/44

US-CL-ISSUED: 395/700
US-CL-CURRENT: 717/170

FIELD-OF-SEARCH: 364/DIG.1, 364/DIG.2, 395/700, 395/650

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4542453	September 1985	Patrick et al.	
4610000	September 1986	Lee	365/189
4688173	August 1987	Mitarai et al.	364/405
4751703	June 1988	Picon et al.	371/10

h e b b g e e e f e e ef b e

4802119	January 1989	Heene et al.	364/900
4831517	May 1989	Crouse et al.	364/200
4982360	January 1991	Johnson et al.	364/900

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
15940	September 1992	WO	
00633	January 1993	WO	

OTHER PUBLICATIONS

No Author, IBM Technical Disclosure Bulletin, vol. 31, No. 1, Jun. 1988, pp. 294-298, "Dual Indirect RAM/ROM Jumptables for Firmware Updates" see whole document.
No Author, IBM Technical Disclosure Bulletin, vol. 35, No. 7, Dec. 1992, pp. 8-13, "Method and Mechanism for Dynamic Loader" see p. 11 line 41, p. 13 line 32.
Bradley et al., IBM Technical Disclosure Bulletin, vol. 27, No. 4A, Sep. 1984, pp. 2187-2188, "Method of Customizing Patches for Each Hardware Configuration" see whole doucument.

ART-UNIT: 236

PRIMARY-EXAMINER: Kriess; Kevin A.

ATTY-AGENT-FIRM: Blakely, Sokoloff, Taylor & Zafman

ABSTRACT:

A method and apparatus for generating patching resources in an information processing system having operating instructions on a Read Only Memory Device. The present invention simplifies the patch generation and installation processes. A patch resource is generated and used by a patch installation process. Patch resources are generated for each ROM version by comparing previous ROM versions to the new ROM version. A patch resource is comprised of a plurality of entries, each of which defines a vector table address, an offset into the vector table and the routine to be inserted. By comparing routines between the ROM versions, routines which are different or new are identified. These routines will become patch resource entries. For patch installation, the ROM version number for the installed ROM is determined; the proper patching resource is retrieved, and the patch resource entries cause the patches to be installed. Patch installation is performed by the steps of modifying vector tables to include the addresses for the new routines.

18 Claims, 8 Drawing figures

☐ 25. Document ID: US 5463775 A

L11: Entry 25 of 27 File: USPT Oct 31, 1995

US-PAT-NO: 5463775
DOCUMENT-IDENTIFIER: US 5463775 A

h e b b g e e e f e e ef b e

TITLE: System and method for performing monitoring of resources in a data processing system in real time

DATE-ISSUED: October 31, 1995

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
DeWitt; Jimmie E.	Georgetown	TX		
Holck; Timothy M.	Austin	TX		
Summers; James H.	Round Rock	TX		
Emrick; Samuel L.	Austin	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 263153 [PALM]
DATE FILED: June 20, 1994

PARENT-CASE:
This is a continuation of U.S. application Ser. No. 07/713,484 filed Jun. 10, 1991. Now abandoned.

INT-CL: [06] G06 F 13/00, G06 F 12/00

US-CL-ISSUED: 395/184.01; 364/DIG.1, 364/282.2, 364/282.1, 364/281.3, 364/550, 364/569, 395/427, 395/650, 395/821
US-CL-CURRENT: 702/186; 710/1, 711/100, 719/321

FIELD-OF-SEARCH: 395/600, 395/275, 395/550, 395/425, 395/200, 364/DIG.1

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4019040	April 1977	Thompson	235/92
4291376	September 1981	McCahill	364/483
4296727	October 1981	Bryan	126/116
4373184	February 1983	Lambregts	364/434
4459656	July 1984	Wilder, Jr.	364/200
4485440	November 1984	Duff et al.	364/300
4533910	June 1985	Sukonick et al.	340/721
4533997	August 1985	Furgerson	364/200
4590550	May 1986	Ellert et al.	364/200
4701848	October 1987	Clyde	395/325
4821178	April 1989	Levin et al.	364/200
4823290	April 1989	Fasack et al.	364/550
4878183	October 1989	Ewart	364/227

h e b b g e e e f e e ef b e

4905171	February 1990	Kiel et al.	364/551.01
4937743	June 1990	Rassman et al.	364/401
5081577	January 1992	Hatle	364/200
5086386	February 1992	Islam	395/600
5153837	October 1992	Shaffer	364/464.04
5214772	May 1993	Weinberger	395/575
5265252	November 1993	Rawson et al.	395/700
5359713	October 1994	Moran et al.	395/200

OTHER PUBLICATIONS

"Monitoring Distributed Embedded Systems", by Ray Ford 1990, pp. 237-244.
"General Purpose Data Collection Method", IBM TDB, vol. 16, No. 6, pp. 1796-1798, Nov. 1973.
"Structure of Performance Monitor for Distributed Computer Systems", IBM TDB, bol. 20, No. 11B, Apr. 1978, pp. 5060-5065.
"Computer with Integraal Function Monitor", IBM TDB, vol. 10, No. 11, Apr. 1968, pp. 1700-1703.
"Performance Evaluator for Operating System", IBM TDB, vol. 16, No. 1, Jun. 1973, pp. 110-118.
"General Trace Facility", IBM TDB, vol. 15, No. 8, Jan. 1972, pp. 2446-2448.
"Definition and Measurement Method of `Working Set` When Analyzing Memory Utilization in OS/s," IBM TDB, vol. 33, No. 2, Jul. 1990, p. 186.
"Software Tool for Reducing Page Fault Frequency", IBM TDB, vol. 32, No. 5B, Oct. 1989, pp. 464-466.
"Performance Monitor of Small Remote Computer Systems", IBM TDB vol. 19, No. 6, Nov. 1976, pp. 2386-2388.
"Memory Usage Estimator", IBM TDB, vol. 16, No. 1, Jun. 1973, pp. 284-285.
"Functional Working Sets", IBM TDB, vol. 19, No. 4, Sep. 1976, pp. 1363-1364.
"Working Set Determination", vol. 26, No. 9, Feg. 1984, p. 4761.
"Working-Set Coprocessor", IBM TDB, vol. 32, No. 4A, Sep. 1989, pp. 240-242.
"Memory Utilization Monitor", IBM TDB, vol. 15, No. 5, Oct. 1972, pp. 1709-1712.
"The OSRM2 System", OS/2 Resource Monitor System, Version 1.1, copyright 1989-1990, C.O.L. Consulting Ltd.

ART-UNIT: 237

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Homere; Jean R.

ATTY-AGENT-FIRM: Bailey; Wayne P. McBurney; Mark E. Henkler; Richard A.

ABSTRACT:

A graphical system resource monitor is provided to depict, in real-time, a data processing system's internal resource utilization. A window or viewport of a data processing system displays user specified internal system resources, such as memory, CPU, or peripheral device availability/utilization. This graphical representation of the `state` of the data processing system's resources is maintained in real-time, while the impact on the system's performance in providing such information is kept to a minimum. This is accomplished through a combination of various techniques, including specialized device drivers for the respective devices coupled with a unique data reduction technique. The graphical results of these resource monitors are continually updated in real-time. This real-time support provides an immediate and accurate representation of the internal operations of the data processing system. Further, these resources can monitored at

h e b b g e e e f e e ef b e

the process level of a multiprocessing system. These representations can be used by a user to identify, isolate, and fine-tune the data processing system's resources to improve the overall efficiency of the system being monitored.

8 Claims, 18 Drawing figures

☐ 26. Document ID: US 5421012 A

L11: Entry 26 of 27 File: USPT May 30, 1995

US-PAT-NO: 5421012
DOCUMENT-IDENTIFIER: US 5421012 A

TITLE: Multitasking computer system for integrating the operation of different application programs which manipulate data objects of different types

DATE-ISSUED: May 30, 1995

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Khoyi; Dana	Dracut	MA		
Soucie; Marc S.	Tyngsboro	MA		
Surppenant; Carolyn E.	Dracut	MA		
Stern; Laura O.	Woburn	MA		
Pham; Ly-Huong T.	Chelmsford	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Wang Laboratories, Inc.	Lowell	MA			02

DISCLAIMER DATE: 20100427

APPL-NO: 08/ 066688 [PALM]
DATE FILED: May 20, 1993

PARENT-CASE:
CROSS REFERENCE TO RELATED APPLICATIONS The present application is a continuation of U.S. patent application Ser. No. 07/938,928, film on Aug. 31, 1992 which issued on Jul. 6, 1993 as U.S. Pat. No. 5,226,161 entitled INTEGRATION OF DATA BETWEEN TYPED DATA STRUCTURES BY MUTUAL DIRECT INVOCATION BETWEEN DATA MANAGERS CORRESPONDING TO DATA TYPES, which is a continuation of U.S. patent application Ser. No. 07/681,435, filed on Apr. 3, 1991 which issued on Apr. 27, 1993 as U.S. Pat. No. 5,206,951 entitled INTEGRATION OF DATA BETWEEN TYPED OBJECTS BY MUTUAL, DIRECT INVOCATION BETWEEN OBJECT MANAGERS CORRESPONDING TO OBJECT TYPES, which is a continuation of U.S. patent application Ser. No. 07/088,622, filed on Aug. 21, 1987, now abandoned. The present application is also related to U.S. patent application Ser. No. 08/123,819, filed on Sep. 20, 1993, which is a continuation of U.S. patent application Ser. No. 07/937,911, filed on Aug. 28, 1992, now abandoned, which is a divisional application of above referenced U.S. patent application Ser. No. 07/681,435, now U.S. Pat. No. 5,206,951. The present application is also related to U.S. patent application Ser. No. 07/936,980, filed on Aug. 28, 1992

h e b b g e e e f e e ef b e

which issued on Nov. 9, 1993 as U.S. Pat. No. 5,261,080 entitled MATCHMAKER FOR ASSISTING AND EXECUTING THE PROVIDING AND CONVERSION OF DATA BETWEEN OBJECTS IN A DATA PROCESSING SYSTEM STORING DATA IN TYPED OBJECTS HAVING DIFFERENT DATA FORMATS, which is a divisional application of above referenced U.S. patent application Ser. No. 07/681,435, now U.S. Pat. No. 5,206,951. The present application is additionally related to U.S. patent application Ser. No. 08/127,981, filed on Sep. 27, 1993 which is a continuation of U.S. patent application Ser. No. 07/915,775, filed on Jul. 16, 1992, now abandoned, which is a continuation of U.S. patent application Ser. No. 07/088,176, filed on Aug. 21, 1987, now abandoned.

INT-CL: [06] G06 F 15/82

US-CL-ISSUED: 395/650; 395/800, 364/DIG.1
US-CL-CURRENT: 718/107; 345/700

FIELD-OF-SEARCH: 395/650, 395/800

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4587628	May 1986	Archer et al.	
4815029	March 1989	Barker et al.	

OTHER PUBLICATIONS

Byte, Aug. 1981, "The Smalltalk-80 System," The Xerox Learning Research Group, pp. 36-48.
Robson, David, "Object-Oriented Software Systems," Byte, Aug. 1981, pp. 74, 76, 78, 80, 82, and 86.
Krasner, Glenn, "The Smalltalk-80 Virtual Machine," Byte, Aug. 1981, pp. 300, 302 304, 306, 308, 310, 312, 314, 316-318, and 320.
Tesler, Larry, "The Smalltalk Environment," Byte, Aug. 1981, pp. 90, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 132, 134, 138, 140, 142, 144, AND 147.
Lipkie, et al., "Stargraphics: An Object--Oriented Implementation," Computergraphics, V. 16, No. 3, Jul. 1982, pp. 29-38.
Schmucker, "Macapp: An Application Framework," Byte, Aug. 1986, pp. 189-193.
Kimura, "A Structure Editor for Abstract Document Objects," IEEE Transactions on Software Engineering, vol. SE-12, No. 3, Mar. 1986, pp. 417-435.
Ursino, "Open Architecture Design Unites Diverse Systems," Electronics, Aug. 11, 1983, pp. 116-117.
Garrett, "Intermedia: Issues, Strategies, and Tactics in the Design of a Hypermedia Document System", Institute for Research in Information and Scholarship (IRIS), Brown University.
Interleaf, Technical Publishing Software, Reference Manual, vol. 1, Sun/Release 3.0, Dec. 1986.

ART-UNIT: 235

PRIMARY-EXAMINER: Lall; Parshotam S.

ASSISTANT-EXAMINER: Ellis; Richard Lee

ATTY-AGENT-FIRM: Milik; Kenneth L.

h e b b g e e e f e e ef b e

ABSTRACT:

An object based data processing system including an extensible set of object types and a corresponding set of "object managers" wherein each object manager is a program for operating with the data stored in a corresponding type of object. The object managers in general support at least a standard set of operations. Any program can effect performance of these standard operations on objects of any type by making an "invocation" request. In response to an invocation request, object management services (which are available to all object managers) identifies and invokes an object manager that is suitable for performing the requested operation on the specified type of data. A mechanism is provided for linking data from one object into another object. A object catalog includes both information about objects and about links between objects. Data interchange services are provided for communicating data between objects of different types, using a set of standard data interchange formats. A matchmaker facility permits two processes that are to cooperate in a data interchange operation identify each other and to identify data formats they have in common. A facility is provided for managing shared data "resources". Customized versions of resources can be created and co-exist with standard resources. A resource retrieval function determines whether a customized or a standard resource is to be returned in response to each request for a resource.

51 Claims, 13 Drawing figures

☐ 27. Document ID: US 5148154 A

L11: Entry 27 of 27 File: USPT Sep 15, 1992

US-PAT-NO: 5148154
DOCUMENT-IDENTIFIER: US 5148154 A

TITLE: Multi-dimensional user interface

DATE-ISSUED: September 15, 1992

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
MacKay; Michael T.	Vallejo	CA		
Berger; Robert J.	Menlo Park	CA		
Duffy; Robert	Milpitas	CA		
Langford; Ted E.	Fremont	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Sony Corporation of America	Park Ridge	NJ			02

APPL-NO: 07/ 622821 [PALM]
DATE FILED: December 4, 1990

INT-CL: [05] G09G 3/02

US-CL-ISSUED: 340/712; 340/729, 340/721

h e b b g e e e f e e ef b e

US-CL-CURRENT: 345/782; 345/156, 345/419, 345/427, 345/723, 345/788, 345/848, 715/500.1

FIELD-OF-SEARCH: 340/729, 340/721, 340/712, 340/723, 364/522, 364/521, 395/154, 395/159

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
Re32632	March 1988	Atkinson	340/709
4533910	August 1985	Sukonick et al.	340/721
4555775	November 1985	Pike	364/900
4622545	November 1986	Atkinson	340/747
4697178	September 1987	Heekel	364/522
4746994	May 1988	Ettlinger	360/13
4748618	May 1988	Brown et al.	370/94
4785408	November 1988	Britton et al.	364/513
4812834	March 1989	Wells	340/721
4847604	July 1989	Doyle	340/706
4868766	August 1989	Oosterholt	364/522
4879751	November 1989	Franks et al.	381/119
4884223	November 1989	Ingle et al.	364/550
4899136	February 1990	Beard et al.	340/706
4914732	April 1990	Henderson et al.	340/825
4931783	June 1990	Atkinson	340/710
4935865	June 1990	Rowe et al.	364/188
4939507	July 1990	Beard et al.	340/706
4994989	February 1991	Usami et al.	340/729
5040131	August 1991	Torres	364/521

OTHER PUBLICATIONS

Alexander, "Visualizing Cleared-Off Desktops" Computer World, May 6, 1991, p. 20.
Video Tape "Sony AVTC Venues and View", Oct. 1990.

ART-UNIT: 269

PRIMARY-EXAMINER: Brier; Jeffery A.

ASSISTANT-EXAMINER: Chin; Jick

ATTY-AGENT-FIRM: Blakely, Sokoloff, Taylor & Zafman

ABSTRACT:

The present invention provides apparatus and methods for a multi-dimensional user interface for use in audio visual production. A display system including a central processing unit (CPU) is coupled through appropriate input/output (I/O) circuitry to input devices, such as a keyboard, digital pad and/or track ball as well as a

h e b b g e e e f e e ef b e

display device. The CPU is further coupled to a hard disk drive for the storage of programs and data, and is also coupled to a network through which the CPU may communicate with a variety of system resource devices such as editors, music synthesizers, graphics generators, scheduling resources, audio enhancement resources, etc. A user viewing the interface on the display may utilize one of the input devices, such as by way of example, the keyboard, to select, incorporate or otherwise integrate the various system resources to develop a unified multi-media production. The user interface of the present invention includes a control frame which in practice substantially fills all of the display screen of the display and is consistent for all user applications. The control frame is comprised of control panels which surround a variety of subwindows and acts as a consistent control area for all users of the interface. Once defined, elements may be selectively placed on an event horizon bar in the control frame. The placement of an element on the event horizon results in the display of timing data for the element, relative to other elements on the event horizon.

56 Claims, 9 Drawing figures

Term	Documents
(9 AND 10).USPT.	27
(L9 AND L10).USPT.	27

h e b b g e e e f e e ef b e

First Hit Fwd Refs
End of Result Set

☐ Generate Collection Print

L3: Entry 1 of 1 File: USPT Jun 26, 2001

DOCUMENT-IDENTIFIER: US 6253225 B1
TITLE: Process executing method and resource accessing method in computer system

Brief Summary Text (13):
The resource management data 105 is shown in FIG. 2A while the process management data 107 is shown in FIG. 2B. As can be seen in FIG. 2B, the process management data 107 is composed of a process identifier 207, a flag 208 indicating presence or absence of the next process management data, and a pointer 209 to the next process management data. At the end of the user application, it becomes necessary to know the identifier of the process having the access right in order to deallocate the resource occupied by the process. Thus, the operating system is provided with a process-specific resource identifier list 109 in which a resource identifier 110 is entered every time when a corresponding resource is generated or every time the access right to the shared resource is acquired.

Brief Summary Text (61):
For achieving the above and other objects which will become apparent as description proceeds, it is taught according to an aspect of the present invention that an identifier composed of address information and generation identifying information of a resource is assigned to a resource upon generation thereof and at the same time the generation identifying information is stored at a leading location of the resource. The generation identifying information is extracted from an identifier transferred as an argument of a system call issued by a user application upon making access to the resource. The extracted generation identifying information is compared with the generation identifying information stored in the resource at the leading location thereof. Access to the resource is enabled when coincidence is found between both the generation identifying information while disabled when discrepancy is found between both the generation identifying information. In this way, the access right to the resource can be controlled solely by comparing both the generation identifying information.

Drawing Description Text (23):
FIG. 20 is a flow chart for illustrating a generation identifying information creating (make_idinf) processing;

Detailed Description Text (12):
Referring to FIG. 6, the interval or period during which the abort-disabled flag 12 remains in the ON-state, i.e., the period from a time point at which the abort disable function 21 is called to set the abort-disabled flag 12 to the ON-state to a time point at which the disabled-abort clear function 22 is called to set the abort-disabled flag 12 to the OFF-state represents the abort-disabled interval. The abort-disabled flag 12 is in the OFF-state when the process is created or generated and set to the ON-state only when the abort disable function 21 is called. In case the forcive termination of the process 200 occurs during the abort-disabled interval during which the abort-disabled flag 12 remains in the ON-state as set by the abort disable function 21 called by the process 200, the process management module 20 executes only the processing for setting the abort request flag 11 to the ON-state while allowing the processing of the process 200 to be continued. The

forcive termination of the process 200 is validated when the abort-disabled flag 12 is set to the OFF-state by the disabled-abort clear function 22 called by the process 200.

Detailed Description Text (46):
In FIG. 12, reference numerals 210 to 212 denote, respectively, those processes which are requesting use of the shared resource, and numeral 220 denotes the sole process that is authorized to perform the processing for a specific shared resource. This process 220 will hereinafter be called the serializer process. The serializer process 220 is resident within the system. A numeral 300 denotes a processing request messaging module for messaging to the serializer process 220 the processing requests issued by the processes 210 to 212, respectively. In correspondence to the serializer process 220, the sole processing request messaging module 300 is provided and constantly resident within the system as in the case of the process 220. Reference numerals 310 to 312 denote, respectively, processing completion messaging modules for transferring the process completion message from the serializer process 220 to the processes 210 to 212, wherein the processing completion messaging modules 310 to 312 are assigned to the processes 210 to 212, respectively, upon generation thereof or upon issuance of the processing request therefrom. The processing request messaging module 300 and the processing completion messaging modules 310 to 312 have internally queues 400, 410 to 412, respectively.

Detailed Description Text (60):
An resource identifier 600 shown in FIG. 17A is an argument of 64 bits contained in the system call issued by a user application for using the same in the access to the resource. Of the 64-bit argument, the leading 32 bits are used for identifying a starting or leading address 601 of a resource 608 while the trailing 32 bits constitute generation identifying information 602. The resource identifier 600 is generated upon creation of the resource and used for making access to the resource until the resource is released or deallocated.

Detailed Description Text (61):
A resource generation counter 603 shown in FIG. 17B is created for each of the processes upon generation thereof and serves as a counter for recording the number of times the resource is generated. The count value 604 of the resource generation counter 603 is set to the initial value of zero and incremented by one every time the resource is created.

Detailed Description Text (62):
Generation identifying information 605 shown in FIG. 17C contains leading 16 bits representing a count value 606 of the resource creation counter with the trailing 16 bits representing a process identifier 607.

Detailed Description Text (63):
Hereinafter, the resource which contains the generation identifying information 609 at a leading or starting location thereof will be referred to as the resource 608.

Detailed Description Text (66):
(1) In a step 2200, a user application issues a system call requesting a process generating processing to the operating system, which responds thereto by creating a resource generation counter 603 for the process as generated.

Detailed Description Text (72):
FIG. 19 shows a flow chart for illustrating the resource allocation processing step 2201 shown in FIG. 18. The resource generation counter is incremented by one (step 2300 in FIG. 19), wherein generation identifying information creating or making processing "make_idinf" is executed (step 2301), which is then followed by step 2302 in which a resource area allocation (or memory area acquisition) processing "alloc_mem" is executed and step 2303 of creating an identifier "make_id" is

executed. Thereafter, the generation identifying information is stored at leading 32 bits of the resource area as returned (step 2304).

Detailed Description Text (73):
.sctn.2.1.1. Generation Identifying Information Creating (make_idinf) Processing

Detailed Description Text (74):
FIG. 20 shows a flow chart for illustrating the generation identifying information creating processing in step 2301 shown in FIG. 19. Hereinafter, this processing will also be referred to as "make_idinf" processing. The generation identifying information is created by a pair of the resource generation counter value and the process identifier of the process destined for the resource generation. Referring to FIG. 20, the process identifier is first acquired (step 2400), wherein the resource generation counter value is placed in the leading 16 bits while a random number value as generated is stored in the trailing 16 bits, to thereby create or make the 32-bit generation identifying information (step 2401). The generation identifying information as created is returned (step 2402).

Detailed Description Text (76):
FIG. 21 shows a flow chart for illustrating the resource acquisition (or allocation) processing in the step 2302 shown in FIG. 19. This processing will also be referred to as "alloc_mem" processing. Because the resource area is constituted by a generation identifying information area and a resource area, the resource area size added with the generation identifying information data size of 32 bits is determined (step 2500), wherein the resource area of the size as determined is allocated (step 2501). Thereafter, the leading address of the resource area (memory area) is returned (step 2502).

Detailed Description Text (78):
FIG. 22 shows a flow chart for illustrating the identifier creating processing in the step 2303 shown in FIG. 19. Hereinafter, this processing will also be referred to as "make_id" processing. The identifier is constituted by a pair of the resource address and the generation identifying information. The address of the resource acquired in resource area allocation or "alloc_mem" processing (step 2302) is placed at leading 32 bits while the generation identifying information created in the generation identifying information creating "make_id" processing (step 2301) is placed at the trailing 32 bits, to thereby create the identifier of 64 bits (step 2600). The identifier as created is returned as the argument for use in making access to the resource (step 2601).

Detailed Description Text (80):
FIG. 23 shows a flow chart for illustrating a resource access processing corresponding to the step 2202 in FIG. 18. Referring to FIG. 23, the leading 32 bits of the identifier received as the argument of the system call issued by the user application are extracted to thereby define the resource address while the trailing 32 bits are extracted for defining the generation identifying information S1 (step 2700). Subsequently, the leading 32 bits are read out from the resource on the basis of the resource address to define the generation identifying information S2, i.e., "read_mem" information (step 2701), wherein the generation identifying information S1 and S2 as extracted are compared with each other (step 2702). Unless the comparison results in coincidence, the access to the resource is not performed, and an error message is returned (step 2703). On the other hand, when the above comparison results in coincidence, then the access processing to the resource is performed (step 2704).

Detailed Description Text (83):
Similarly to the resource access processing (step 2202 shown in FIG. 18), in the resource deallocation processing (step 2203 shown in FIG. 18), the leading 32 bits of the identifier received as the argument of the system call issued by the user application are extracted to define the resource address while the trailing 32 bits

are extracted for defining the generation identifying information S1 (step 2800). Subsequently, the leading 32 bits are read out from the resource to define the generation identifying information S2 "read_mem" processing (step 2801). Then, the generation identifying information S1 and S2 mentioned above are compared with each other (step 2802). Unless the comparison results in coincidence, the resource is not deallocated and a corresponding error message is returned (step 2803). On the other hand, when the above comparison results in coincidence, then the resource deallocation processing is performed (step 2804).

Detailed Description Text (87):
By contrast, according to the methods of the invention, the address of the resource can be obtained for unlocking the resource without need for following the pointers for acquiring the address of the resource because the address has already been contained in the identifier of the resource X to be deallocated. Additionally, it is apparent from the previous description concerning the resource deallocation processing (.sctn.2.3.), the operating system controls or manage the resource access right on the basis of only the result of comparison between the generation identifying information S1 contained in the identifier assigned to the resource X and the generation identifying information S2 stored in the leading of the resource X. Thus, the task imposed on the operating system is only the deallocation of the resource X.

Detailed Description Text (89):
(3) In conjunction with the resource deallocation processing described above, it is again assumed that a process C generates a resource Y and allocates the created resource Y to the address having been allocated to the resource X in the state in which the resource X is deallocated or released. In that case, the operating system issues to the process B no information or message indicating that the resource X has been deallocated. Consequently, there may arise such situation that the process B tries to make access to the resource Y on the basis of the address information contained in the identifier of the process B. However, because the identifier contains not only the address information but also the generation identifying information and because the generation identifying information is different between the identifier assigned to the resource X and the identifier assigned to the resource Y, the process B trying to make access to the resource Y with its own identifier can not access the resource Y. In this manner, the illegal access can be positively inhibited or disabled.

Detailed Description Text (93):
Upon starting-up of the system, a 32-bit system clock is generated for holding the time as lapsed on a 10-.mu. second basis.

Detailed Description Text (94):
When a resource is created, the value of the system clock is fetched as the generation identifying information which is then stored at a leading address of the resource. An identifier of 64 bits is created which is composed of 32 MSB (more significant bits) corresponding to the address of the resource and 32 LSB (less significant bits) corresponding to the generation identifying information.

Detailed Description Text (95):
For making access to the resource, the generation identifying information is extracted from the identifier transferred as the argument of the system call from the user application to the operating system, wherein the extracted generation identifying information is compared with the generation identifying information stored at the leading or starting address of the resource. When this comparison results in coincidence, the access to the resource is enabled, i.e., allowed. On the contrary, when the comparison shows discrepancy, the access to the resource is disabled or inhibited with an error message being returned.

Detailed Description Text (97):

When the system is activated, a single 32-bit counter is generated in the system for counting the number of times which resources are generated, and an initial value "0" is inputted to the counter as the generation identifying information.

Detailed Description Text (98):
Upon creation of a resource, the above-mentioned generation identifying information is incremented by one and stored at the starting or leading address of the resource. Subsequently, a 64-bit identifier is created which includes more significant 32 bits corresponding to the address of the resource and less significant 32 bits corresponding to the generation identifying information.

Detailed Description Text (103):
When the processings 1302 and 1306 shown in FIG. 7 are executed for carrying out the shared resource access processing, processing illustrated in FIG. 23 is executed for validating the access to the shared resource. At first, before making access to the shared resource, the generation identifying information (602 in FIG. 17A) stored in the identifier is compared with the generation identifying information (609 in FIG. 17A) of the shared resource (step 2702 in FIG. 23).

Detailed Description Text (104):
When coincidence is found between both the generation identifying information, it is then decided that the identifier of the shared resource is valid, i.e., the resource address (601 in FIG. 17A) contained in the identifier is valid. Accordingly, access is made to the shared resource by using the resource address to perform the processing for the shared resource.

Detailed Description Text (105):
On the contrary, when discrepancy is found between both the generation identifying information mentioned above, this means that the identifier of the shared resource is invalid. More specifically, assuming, by way of example, that discrepancy of the generation identifying information occurs in the processing step 1306 shown in FIG. 7, this means that the shared resource is deallocated for some reason during the preempt-enable interval intervening between the release of the preempt disabled-state (step 1303 in FIG. 7) in succession to the completion of the processing 1302 shown in FIG. 7 and the next preempt disabling (step 1305 in FIG. 7). Thus, the address of the resource (shared resource) as contained in the identifier is invalid. Consequently, the access to the shared resource is suspended and an error processing (step 2703 in FIG. 23) is carried out. In the error processing, the preempt disabling or inhibition and/or abort disabling may be cleared as occasion requires.

First Hit Fwd Refs

End of Result Set

☐ Generate Collection Print

L22: Entry 1 of 1 File: USPT Jun 26, 2001

DOCUMENT-IDENTIFIER: US 6253225 B1
TITLE: Process executing method and resource accessing method in computer system

Abstract Text (1):
A process executing method capable of performing multiprocessing by using a shared resource without impairing periodical drivability of processes designed for executing continuous media processing. When a process requests the use of the shared resource, abortion of that process is first disabled by the process itself by using an abort disable function and then preemption of the same process is disabled by means of a preempt control module, whereupon the process enters a processing executed by using the shared resource. Upon completion of the processing for the shared resource, the process is immediately set to a preempt-enabled state by means of a preempt control module. After completion of all the processings, the abort-disabled state is finally cleared by using a disabled-abort clear function. Upon occurrence of forcive termination of a process in the abort-disabled state thereof, execution of this process is continued until the abort-disabled state is cleared, and the process is terminated forcibly after the abort-disabled state has been cleared.

Brief Summary Text (7):
On the other hand, the operating system providing the multiprocessing environment is imparted with a function for protecting the resource allocated to a given process against the illegal access attempted by any other process. This function is referred to as the access right control or manage function. The access right control or management now under consideration is based on the presumption that the access right can be set on a process-by-process basis and that a plurality of processes may simultaneously try to access a computer resource. An interface for allowing a user application to call the functions of the operating system is ordinarily provided and known as "system call". In this conjunction, it is noted that when a pointer is used as an argument of the system call, designation of an illegal address by the user application may unfavorably lead to destruction of important data resident in the operating system. To avoid such unwanted situation, an identifier is used as the argument in place of the pointer in typical one of the access right control or management.

Brief Summary Text (8):
In the case of the access right control method in which the identifier is used as the argument in the system call issued by the user application, the operating system is required to translate the identifier to an address in the resource for making access to the resource. In this context, the simplest one of the translation methods is a method in which the hash function is employed. According to this method, an identifier is placed in or assigned to the hash function as a key, whereon the hash value as obtained is used as the address. The access right control method relying on the hash function will be described below.

Brief Summary Text (10):
As is shown in FIG. 1, the operating system assigns a resource identifier 100 to the hash function (F) 101 as a key to acquire as the hash value an index "Index"

102 contained in a resource management data table 104. As is shown in FIG. 2A, resource management data 105 stored in the resource management data table 104 contains an identifier 201, a pointer 205 to a resource 103, a flag 202 indicating presence or absence of a succeeding or next resource management data 106, a pointer 203 to the succeeding resource management data 106, and a pointer 204 to process management data 107.

Brief Summary Text (11):
There may arise such situation that the hash function returns a same index "Index" for different identifiers RID. In that case, collision between the indexes "index" will occur. Accordingly, the second resource management data 106 et seq. are stored in an overflow area with the address of the second or succeeding resource management data 106 being placed in the immediately preceding resource management data 105.

Brief Summary Text (12):
In case the resource is a shared resource, there exist a plurality of processes having respective access rights for the same identifier RID. In that case, the pointers to the second process identifier data 108 et seq. are placed in the immediately preceding process management data 107.

Brief Summary Text (13):
The resource management data 105 is shown in FIG. 2A while the process management data 107 is shown in FIG. 2B. As can be seen in FIG. 2B, the process management data 107 is composed of a process identifier 207, a flag 208 indicating presence or absence of the next process management data, and a pointer 209 to the next process management data. At the end of the user application, it becomes necessary to know the identifier of the process having the access right in order to deallocate the resource occupied by the process. Thus, the operating system is provided with a process-specific resource identifier list 109 in which a resource identifier 110 is entered every time when a corresponding resource is generated or every time the access right to the shared resource is acquired.

Brief Summary Text (16):
When the system call requesting the resource allocation is issued to the operating system by a user application, the operating system responds thereto by allocating the resource (step 1000). Subsequently, the operating system acquires an identifier RIDx definite in the system (step 1001) and places the identifier RIDx to the hash function as a key. Thus, the index "Index" contained in the resource management data table 104, i.e., the address where the resource management data x is stored is obtained (step 1002). In the case where the resource management data y has already been placed at the leading location of the area designated by the index, i.e., when collision between the indexes occurs (step 1003), the pointers are followed up from one to another (step 1005) so long as the flag 202 indicating presence or absence of the identifier management data 105 assumes a value "ON" (indicating the presence of the resource management data 105) (step 1004). When the resource management data z for which the flag 202 indicating the presence or absence of the succeeding identifier management data is set to "OFF" is found in the course of following or tracing the pointers, then the succeeding identifier management data presence/absence flag 202 is set to "ON" (step 1006). Subsequently, an area for the new resource management data x is allocated to the overflow area (step 1007), whereupon the pointer 203 to the immediately preceding resource management data contained in the resource management data y is placed at the address of the resource management data x (step 1008). When it is found in step 1003 that the succeeding identifier management data is not stored, the area for the resource management data x is assigned to the main area (step 1009).

Brief Summary Text (17):
The process management data 107 is placed in the process management area and the process identifier is stored therein (step 1010). At the same time, the identifier

RIDx 207 is stored in the process-specific identifier list 109 (step 1011). The identifier RIDx 201, the pointer 204 to the process management data and the resource pointer 205 contained in the resource management data x 105 are assigned with respective values, whereon the flag 202 indicating the presence or absence of the succeeding resource management data is set to "OFF" state (step 1012). Then, the identifier RIDx is returned to the user application (step 1013).

Brief Summary Text (19):
When the system call indicating a resource access request is issued by the user application to the operating system, the latter executes the processing illustrated in FIG. 4.

Brief Summary Text (20):
The operating system assigns the identifier RIDx which is the argument of the system call to the hash function to obtain the index of the resource management data 105 (step 1100). Subsequently, the identifier RID contained in the resource management data 105 located at the index is compared with the argument RIDx (step 1101). When the values of the identifier RID and the argument RIDx differ from each other and when the flag 202 indicating the presence or absence of the succeeding resource management data in the resource management data 105 is set "ON", i.e., when the succeeding resource management data 105 exists (step 1102), the pointer 203 is followed up to the succeeding resource management data (step 1103), wherein the comparison between the identifier and the argument mentioned above is again performed (step 1101). In the case where the identifiers RID of all the resource management data 105 which can be followed with the pointers do not coincide with the argument RIDx, it is then decided that the resource as requested does not exist in the system whereupon an error message is returned to the user application (step 1104). On the other hand, when coincidence is found between the identifier RID contained in the resource management data 105 and the argument RIDx in the comparison step 1101, then the process identifier PIDx of the process accessing currently the resource is compared with the process identifier PID 207 contained in the process management data 107 (step 1105). When it is found in step 1105 that the values of both the identifiers differ from each other and when the flag 208 contained in the process management data 107 and indicating the presence or absence of the succeeding process management data is "ON" (indicating that the succeeding process management data 107 exists) (step 1106), the succeeding pointer 209 is followed to reach the succeeding process management data (step 1107), whereon the comparison of the process identifiers mentioned above is again performed (step 1105). When the process identifiers PID contained in all the resource management data 105 which can be followed with the aid of the pointers do not coincide with the argument PIDx, it is then decided that the user application issuing the system call has no access right to the resource, wherein an error message is returned to the user application (step 1108).

Brief Summary Text (21):
On the other hand, when the coincidence is found between the process identifier and the argument in step 1105, access to the resource is performed by using the resource address stored in the resource pointer 205 contained in the resource management data 105 (step 1109).

Brief Summary Text (23):
In the case where a system call requesting the deallocation of the resource is issued to the operating system from the user application, then the resource deallocation processing is performed for that resource. Additionally, when deallocation of the resources becomes necessary due to abnormal termination of a process, then the resource deallocation processing is performed for the resources corresponding to all the identifiers contained in the identifier list specific to the process terminated abnormally. The resource deallocation processing will be described below by reference to FIG. 5. In the first place, the operating system inhibits or block the access to all the resources (step 1200) and places the

identifier RIDx of the resource to be deallocated in the hash function as a key to obtain the index "Index" of the resource management data 105 (step 1201). Subsequently, the identifier RID contained in the resource management data 105 stored in the area indicated by the index is compared with the key or identifier RIDx (step 1202). When the values of both the identifiers differ from each other and when the flag 202 indicating the presence or absence of the succeeding resource management data is "ON", i.e., when the succeeding resource management data 105 exists (step 1203), the pointer 203 is followed up to the succeeding resource management data (step 1204), whereon the identifier comparison mentioned above is again repeated (step 1202). In case the identifiers RID of all the resource management data 105 which can be followed with the pointers do not coincide with the key or identifier RIDx, it is then decided that the resource as requested does not exist, whereupon an error message is returned to the user application (step 1205).

Brief Summary Text (24):
On the other hand, when coincidence between both the identifiers RID and RIDx is found in the step 1202, the address of the process management data 107 stored in the pointer 204 to the process management data is acquired (step 1206). When the succeeding resource management data 105 exists (step 1207), the resource management data 105 specified by the identifier RIDx is released or unlocked (step 1209) after changing the string of the pointers 203 to the resource management data (step 1208). Subsequently, the pointer acquired in step 1206 is followed and it is checked whether or not the flag contained in the process management data 107 and indicating the presence or absence of the succeeding process management data is "ON" (step 1210). If the flag is "ON", the pointer 209 to the succeeding process management data is acquired (step 1211), whereupon a message indicating that the process management data 107 is released or deallocated and that the identifier RID can no more used is issued to the user applications, i.e., individual processes (step 1212). On the other hand, when it is dedicated in step 1210 that the flag 208 indicating the presence or absence of the succeeding resource management data is "OFF", the final process management data 107 is released or deallocated, wherein the message indicating that the identifier RID can no longer be used is issued to the user application (step 1213), which is then followed by the reopening of the access to all the resources (step 1214).

Brief Summary Text (25):
In conjunction with the method described above, it is noted that even when the index "Index" contained in the resource management data table 104 can be obtained by assigning an illegal identifier RIDz to the hash function as the key, the illegal identifier RIDz can not be found in the resource management data 105 which can be followed or traced with the index. Consequently, no resource address can be obtained, rendering the access impossible. Further, even if the illegal identifier RIDz should be contained accidentally in the resource management data 105 traced with the index, the identifier of the process making access to the resource is not stored in the process management data 107. Consequently, the address of the resource can not be gained. Thus, illegal access can be inhibited or disabled.

Brief Summary Text (26):
Furthermore, in the resource unlock processing, the resource protection can be realized by inhibiting or disabling all the accesses of the user applications to the resource, while illegal access after the resource deallocating can be inhibited by invalidating the identifier, releasing the resource management data and the process management data which can be traced with the index and issuing the message informing the processes of the deallocating of the resource.

Brief Summary Text (31):
Under the circumstances, in the system designed for performing the continuous media processing, there is conceivable a method of structuring a computer system without using any lock at all from the beginning in order to evade the priority inversion

problem. As a method of realizing the exclusive control or mutually exclusive management of the process, such process control or management method may be conceived in which the process using currently the shared resource is allowed to occupy exclusively the CPU (Central Processing Unit) or, to say in another way, any other process is inhibited or disabled from being scheduled so long as the shared resource is being used by a given one of the processes. Such control or managing method can be realized by adopting a preempt control method proposed in Japanese Patent Application No. 8-97997. Parenthetically, the preempt control method concerns prevention of a process being executed from being temporarily suspended (i.e., preempted) by providing interfaces "preempt disabling" and "disabled-preempt releasing or clearing", wherein during a period intervening between a time point at which the preempt disabling is validated and a time point at which the preempt disabling is cleared (this period may also be referred to as the preempt-disabled interval), the process in execution is prevented from being preempted and thus can continue the execution even when a scheduling request is issued from any other process. By virtue of such control, it can be ensured that no more than one process can use the shared resource at any time. In other words, the exclusive control or management of the processes can thus be realized.

Brief Summary Text (34):
When the processing mentioned above is to be realized only by setting the preempt-disabled interval, it is then necessary to set the preempt-disabled interval so that it extends from a time point "preceding to contacting the free list" to a time point at which "the resource has been chained to the resource allocation list". To this end, processings (1a) to (5a) mentioned below will have to be carried out.

Brief Summary Text (35):
(1a) Setting the preempt-disabled state.

Brief Summary Text (39):
(5a) Clearing the preempt-disabled state as set.

Brief Summary Text (40):
At this juncture, it is noted that the time taken for initialization of the resource is not always short. In fact, time in the order of several ten milliseconds is taken solely for the initialization of a memory upon memory allocation internally of the operating system. Accordingly, when the mutually exclusive control is to be realized only by setting the preempt-disabled interval, as mentioned above, there may arise such situation that a given one process occupies the CPU for an extended time, which will result in degradation in the response performance on the real-time basis, eventually exerting adverse influence to the periodical drivability of the continuous media processing.

Brief Summary Text (42):
(1b) Setting the preempt-disabled state.

Brief Summary Text (44):
(3b) Clearing the preempt-disabled state as set.

Brief Summary Text (46):
(5b) Setting the preempt-disabled state.

Brief Summary Text (48):
(7b) Clearing the preempt-disabled state as set.

Brief Summary Text (49):
The above processings can certainly satisfy the necessary condition from the view point of the mutual exclusive control or management. Besides, the duration of the preempt-disabled interval can be reduced to ca. 10 .mu.sec., while in the resource initialization processing (4b), the right of using the CPU can be transferred to

other process. Thus, the real-time response performance of the system can be improved. Unfavorably, however, there may arise a problem when the process executing the resource initialize processing (4b) is externally forced to terminate in the course of executing this processing.

Brief Summary Text (50):
In general, most of computer systems are equipped with a function for stopping externally process execution for coping with overrunning of a program. Accordingly, when the process executing the processings mentioned above is forced to terminate by other process by resorting to the forcive terminate function mentioned above, then the resource initialized by the processing (4b) may become a free resource belonging to none of the management lists. As the result of this, such situation may arise in which the operating system can not recover the resource when the process is terminated, which in turn means that the resource used by the process terminated forcibly becomes unusable for ever. In order to evade this problem, there may be conceived a method of initializing the resource in the state where the resource is chained to the free list or the resource allocation list. In that case, however, the duration of the preempt-disabled interval period is substantially equal to that of the preempt-disabled period covering the whole processings described hereinbefore. As another method, it is equally conceivable to prepare separately a list for managing the resources under initialization, wherein the resource undergoing the processing (4b) is chained to this list. However, this method is disadvantageous in that overhead involved in the initialization processing increases because of an increased number of times the list-chain changing processing has to be performed.

Brief Summary Text (53):
According to the teaching of the invention, it is declared by the process in precedence to the use of the shared resource that the process is not forcibly terminated. Parenthetically, in the description which follows, expression "process is not aborted" or the like is used, which means that the process is not forcibly terminated, and prevention of the process from being forcibly terminated is expressed as "abort disabling" or the like. Thus, "the state in which a process is prevented from being aborted" may be expressed as "abort-disabled state" or the like. Additionally, the process is "preempt-disabled", which mean that the process is protected against interruption or suspension of the processing or task executed by that process. At the end of the use of the shared resource, the process is cleared from the preempt-disabled state. Upon completion of all the processings for the shared resource, it is declared by the process that it may be forcibly terminated, which is expressed as "clearing the process from the abort-disabled state" or simply as "clearing of the abort-disabled state" or so. Further, the time period or interval intervening between the abort-disable declaration and the abort-disabled state clearing declaration is called "abort-disabled interval". When a request for the forcive termination of a process is issued during the abort-disabled period or interval, execution of the process is continued until the processing to be executed during the abort-disabled period has been completed. When the abort-disabled state is cleared, the process is forcibly terminated.

Brief Summary Text (54):
In a preferred mode for carrying out the invention, a sole process dedicated for coping with such situation that the shared resource is used by a process over a prolonged duration is provided in the system. Further, for sending the requests for using the shared resource to the sole process, there is provided a queue. The other process destined for performing the continuous media processing issues a shared resource use request to the sole dedicated process before starting the periodical driving, and upon completion of the processing for the shared resource, the periodical driving is validated. The request mentioned above is registered in the queue. Provision of such dedicated process can ensure that it is always only one process in the system that can perform the processing for the shared resource. Thus, the processing for the shared resource can be effectuated in the preempt-

enabled state, which in turn means that the processing for the shared resource can be executed in parallel with the periodically driven process or processes.

Brief Summary Text (56):
On the other hand, in the case of the method according to which the identifier is placed in or assigned to the hash function as a key to determine the index "Index" of the resource management data containing the address of the resource for making access to the resource, there arise problems mentioned below in executing such processing which has to process a large amount of data at a high speed on a real-time basis while allocating a CPU time at every predetermined interval as in the case of the multi-media data processing.

Brief Summary Text (57):
(1) When a plurality of processes make simultaneous access to one data, collision of the hash values will take place in the accessing method using the hash function, which makes it necessary to search the overflow area while following pointers until the identifier of the resource as searched is found.

Brief Summary Text (58):
Further, when the identifier is found, it is necessary to search the process management data by following with the pointers until the process identifier is found in order to check whether or not the process attempting to make access to the shared resource has really the access right (i.e., the right of accessing the resource). Consequently, overhead involved in the processing increases considerably while the time taken for the memory access becomes too long to be useful for effective reproduction of the data.

Brief Summary Text (59):
(2) In the system in which the user application unlocks the resource, the operating system has to follow or trace the pointers of all the process management data in order to acquire the identifier of the process using the resource and release the process management data as found while messaging to the individual processes that the resource can not be used. During the period for the search mentioned above and for the execution of succeeding processings, all the accesses from the user applications to the resource are disabled or inhibited. Consequently, in the case where the resource deallocation processing takes place and in the system in which there exist a plurality of processes to be processed on a real time basis, overhead involved in searching the processes sharing the resource to be deallocated and executing the succeeding deallocation processing will increase considerably. In that case, the real-time processing is very difficult to realize or rendered impossible.

Brief Summary Text (60):
In the light of the foregoing, it is a second object of the present invention to provide an accessing method which is capable of reducing the overhead involved in the translation between the identifier and the address while sustaining the access right control function of the conventional method and decreasing the time or period during which the access to the shared resource for executing the process terminating processing is disabled.

Brief Summary Text (61):
For achieving the above and other objects which will become apparent as description proceeds, it is taught according to an aspect of the present invention that an identifier composed of address information and generation identifying information of a resource is assigned to a resource upon generation thereof and at the same time the generation identifying information is stored at a leading location of the resource. The generation identifying information is extracted from an identifier transferred as an argument of a system call issued by a user application upon making access to the resource. The extracted generation identifying information is compared with the generation identifying information stored in the resource at the

leading location thereof. Access to the resource is enabled when coincidence is found between both the generation identifying information while disabled when discrepancy is found between both the generation identifying information. In this way, the access right to the resource can be controlled solely by comparing both the generation identifying information.

Drawing Description Text (11):
FIG. 8 is a flow chart illustrating a processing procedure of an abort disable function (21) shown in FIG. 6;

Drawing Description Text (12):
FIG. 9 is a flow chart illustrating a processing procedure of a preempt disabling function (32) shown in FIG. 6;

Drawing Description Text (13):
FIG. 10 is a flow chart illustrating a processing procedure of a disabled-preempt clear function (33) shown in FIG. 6;

Drawing Description Text (14):
FIG. 11 is a flow chart illustrating a processing procedure of a disabled-abort clear function (22) shown in FIG. 6;

Drawing Description Text (25):
FIG. 22 is a flow chart for illustrating a identifier creating (make_id) processing;

Detailed Description Text (4):
In the figure, reference numeral 10 denotes a process management table 10 for controlling or managing processes. In the process management table 10, there are held an abort request flag 11, an abort-disabled flag 12, a counter 13 for counting abort disablings nested. Further, reference numeral 20 denotes a process management module for controlling or managing processes, 21 designates an interface function for setting the abort disabling, 22 denotes an interface function for clearing the abort disabling, and 200 denotes a process which requires the processing for a shared resource. Furthermore, reference numeral 40 denotes a function which uses a shared resource provided in the process 200 and which holds work areas 23 and 37. Additionally, reference numeral 30 denotes a preempt control module which is based on the teaching of the invention disclosed in Japanese Patent Application No. 8-97997. More specifically, the preempt control module 30 includes a scheduler 31, a preempt disable function 32, a disabled-preempt clear function 33, a scheduling request flag 34, a preempt-disabled flag 35 and a counter 36 for counting the preempt disablings nested.

Detailed Description Text (5):
FIG. 7 is a flow chart for illustrating a flow of controls for the abort disabling and the preempt disabling in a process for realizing a feature of the present invention. When processing relating to the shared resource is to be executed, the function 40 included in the process 200 destined for executing the above-mentioned processing initially inhibits its own process from being aborted by using the abort disable function 21 (step 1300). Immediately before using the shared resource, the function 40 inhibits its own process from being preempted by using the preempt disable function 32 held in the preempt control module 30 (step 1301), wherein the processing which makes use of the shared resource is executed (step 1302). Upon completion of the processing for the shared resource, the disabled preempt is instantaneously released or cleared with the aid of the disabled-preempt clear function 33 held in the preempt control module 30 (step 1303). At this juncture, it should be mentioned that setting of the preempt-disabled interval (or preempt-disabled section in light of the control flow) extending from the preempt disabling to the disabled-preempt clearing (corresponding to the section extending from step 1301 to step 1303) is limited to such processing for the shared resource which is

terminated within a time exerting no adverse influence to the periodical driving performance, e.g. a time not longer than 10% of a minimum time required for managing the periodical driving. During a period preceding to the subsequent declaration of the preempt disabling, there prevails a preempt-enabled state where processings require relatively a lot of time such as initialization for the allocated resource or the like is executed (step 1304). During this period, execution of the process 200 may be temporarily interrupted (or suspended) while allowing other processes to be scheduled. When the processing for the shared resource becomes necessary again, the process 200 sets itself once more to the preempt-disabled state (step 1305). When the processing for the shared resource (step 1306) comes to an end, the preempt disabling is released or cleared (step 1307). Upon completion of all the processings, the abort disabling is cleared finally by resorting to the use of the disabled-abort clear function 22 (step 1308).

Detailed Description Text (6):
In the case of the example illustrated in FIG. 7, the preempt-disabled interval makes appearance twice during the period extending from the abort disabling to the disabled-abort clearing. In actual applications, the number of such preempt-disabled intervals may be more than three inclusive thereof.

Detailed Description Text (7):
FIG. 8 is a flow chart illustrating in detail an internal processing of the abort disabling procedure (1300) mentioned above by reference to FIG. 7. Execution of the processing shown in FIG. 8 is effectuated by using the abort disable function 21. To this end, the abort disable function 21 checks at first whether the abort-disabled flag 12 held in the process management table 10 managing the process 200 is in the OFF-state or not (step 1400). When the abort-disabled flag 12 is OFF, then the abort disable function 21 sets the abort-disabled flag 12 to the ON-state (step 1401). Subsequently, the abort disable function 21 increments the value of the counter 13 by one (step 1402). The procedure shown in FIG. 8 is indivisible in execution thereof.

Detailed Description Text (8):
FIG. 9 is a flow chart illustrating in detail internal processing of the preempt disabling procedure (1301, 1305) mentioned above by reference to FIG. 7. The processing shown in FIG. 9 is executed by means of the preempt disable function 32 shown in FIG. 6. At first, the preempt disable function 32 checks whether the preempt-disabled flag 35 held in the preempt control module 30 is in the OFF-state or not (step 1500). When the preempt-disabled flag 35 is OFF, then the preempt disable function 32 sets the preempt-disabled flag 35 to the ON-state (step 1501). Subsequently, the preempt disable function 32 increments the value of the counter 36 by one (step 1502). The whole procedure shown in FIG. 9 is executed indivisibly.

Detailed Description Text (9):
FIG. 10 is a flow chart illustrating in detail internal processing of the disabled-preempt clearing procedure (1303, 1307) mentioned above by reference to FIG. 7. Execution of the processing shown in FIG. 10 is in charge of the disabled-preempt clear function 33 shown in FIG. 6. At first, the disabled-preempt clear function 33 checks whether or not the nest value (described hereinafter) of the function transferred as the first argument coincides with the value of the counter 36 (step 1600). Unless coincidence is found, then an abnormality processing is carried out. On the other hand, when the coincidence is found, the value of the counter 36 incorporated in the preempt control module 30 is decremented by one (step 1601). When the value of the counter 36 resulting from the decrementation is positive (plus), then the processing is terminated intact (step 1602). On the contrary, in case the value of the counter 36 is zero or negative (minus), the preempt-disabled flag is set to the OFF-state (step 1603). Subsequently, the scheduling request flag 34 held internally of the preempt control module 30 is checked (step 1604). When

this flag is OFF, then the processing is terminated. On the other hand, when the scheduling request flag 34 is in the ON-state, the scheduler 31 is activated (step 1605) to thereby validate the execution of the scheduling processing for the process.

Detailed Description Text (10):
FIG. 11 is a flow chart illustrating in detail internal processing of the disabled-abort clearing procedure (1308) mentioned above by reference to FIG. 7. The processing shown in FIG. 11 is executed with the aid of the disabled-abort clear function 22 shown in FIG. 6. At first, the disabled-abort clear function 22 checks whether or not the nest value (described hereinafter) of the function transferred as the first argument coincides with the value of the counter 13 (step 1700). Unless coincidence is found, then an abnormality processing is performed. On the other hand, when coincidence is confirmed, the value of the counter 13 incorporated in the process management table 10 managing the process 200 is decremented by one (step 1701). When the value of the counter 13 resulting from the decrementation is positive (plus), the processing is terminated intact (step 1702). By contrast, in case the value of the counter 13 is zero or negative (minus), the abort-disabled flag is set to the OFF-state (step 1703). Subsequently, the abort request flag 11 held internally of the process management table 10 is checked (step 1704). When this flag is OFF, then the processing is terminated. On the other hand, when the abort request flag 11 is ON, the process management module 20 is activated (step 1705) to thereby validate the execution of the abort processing for the process 200.

Detailed Description Text (12):
Referring to FIG. 6, the interval or period during which the abort-disabled flag 12 remains in the ON-state, i.e., the period from a time point at which the abort disable function 21 is called to set the abort-disabled flag 12 to the ON-state to a time point at which the disabled-abort clear function 22 is called to set the abort-disabled flag 12 to the OFF-state represents the abort-disabled interval. The abort-disabled flag 12 is in the OFF-state when the process is created or generated and set to the ON-state only when the abort disable function 21 is called. In case the forcive termination of the process 200 occurs during the abort-disabled interval during which the abort-disabled flag 12 remains in the ON-state as set by the abort disable function 21 called by the process 200, the process management module 20 executes only the processing for setting the abort request flag 11 to the ON-state while allowing the processing of the process 200 to be continued. The forcive termination of the process 200 is validated when the abort-disabled flag 12 is set to the OFF-state by the disabled-abort clear function 22 called by the process 200.

Detailed Description Text (13):
Further, referring to FIG. 6, the interval or period during which the preempt-disabled flag 35 remains in the ON-state, i.e., the period extending from a time point at which the preempt disable function 32 is called to set the preempt-disabled flag 35 to the ON-state to a time point at which the disabled-preempt clear function 33 is called to set the preempt-disabled flag 35 to the OFF-state represents the preempt-disabled interval. The preempt-disabled flag 35 is in the OFF-state when the system is activated and set to the ON-state only when the preempt disable function 32 is called. In case the scheduling request for the other process(es) occurs during the interval in which the preempt-disabled flag 35 remains in the ON-state as set by the preempt disable function 32 called by the process 200, the scheduler 31 executes only the processing for setting the scheduling request flag 34 to the ON-state while allowing the processing of the process 200 to be continued without scheduling the other process(es). When the preempt-disabled flag 35 is set to the OFF-state by the disabled-preempt clear function 33 called by the process 200, the scheduler 31 sets the scheduling request flag to the OFF-state and suspends the execution of the process 200 while allowing other processes to be scheduled.

Detailed Description Text (14):
Referring to FIG. 7, in the interval or section extending from step 1301 to step 1303 and in the interval or section extending from step 1305 to step 1307, the preemption is disabled, which in turn means that the right of using the CPU (Central Processing Unit) is never transferred to other process. Thus, it is only the running process that can use the shared resource. In other words, any other process is inhibited from using the shared resource, whereby the exclusive control or management of the resource can be realized, which makes it possible to perform the continuous media multiprocessing using the shared resource. Additionally, because abortion is disabled during the interval spanning over steps 1300 to 1308, any process acquired the resource is caused to vanish due to the forcive termination. Thus, the resource is positively prevented from falling in such state that it can not be utilized for an indefinite period.

Detailed Description Text (15):
The abort disabling as well as the disabled-abort clearing is effectuated through the medium of interfaces mentioned below, respectively.

Detailed Description Text (17):
abort_disable(*level)

Detailed Description Text (19):
level: the depth of the nest formed by a pair of the instant function now under consideration and the function "abort_enable" is returned.

Detailed Description Text (21):
The function now of concern can make the currently executed process transit to the abort-disabled state. This function and the function "abort_enable" may be issued in a pair during the section or period spanning over the transition to the abort-disabled state in response to the issuance of this function and the restoration to the abort-enabled state in response to the issuance of the function "abort_enable". This means that the pair of the function now of concern and the function "abort_enable" can be nested. The function now of concern and the function "abort_enable" issued internally of the nest make no state transition between the abort-disabled state and the abort-enabled state. The argument "level" reflects the depth of such nested state.

Detailed Description Text (23):
abort_enable(level)

Detailed Description Text (25):
level: the level of the nest returned from the function "abort_disable" which constitutes the counterpart to be paired with the instant function is designated.

Detailed Description Text (27):
The function of concern can make the currently executed process be restored to the abort-enabled state. The function now of concern (i.e., "abort_enable") and the function "abort_disable" may be issued in a pair during the section or period spanning over the transition to the abort-disabled state in response to the issuance of the function "abort_disable" and the restoration to the abort-enabled state in response to the issuance of the instant function. This means that the pair of the function "abort_disable" and the function now of concern "abort_enable" can be nested. The function "abort_disable" and the function now of concern issued internally of the nest make no state transition between the abort-disabled state and the abort-enabled state. The argument "level" designates or represents the depth of such nest, i.e., the value of "level" due to the issuance of the counterpart function "abort_disable". This value is held by the operating system as well. Thus, discrepancy of this value with the depth of the nest specified by the argument, the so-called error return takes place.

Detailed Description Text (28):
When the function 40 included in the process 200 uses the abort disable function 21, the function 40 stores in its own work area 23 the current depth of the nest returned from the abort disable function 21. For issuing the disabled-abort clear function 22, the value stored in the work area 23 is designated or specified as the argument of the function 40. The disabled-abort clear function serves to compare the value mentioned above with the current depth of the nest to validate the error return unless coincidence is found between both the values mentioned above. By virtue of this function, it is possible to detect easily such a programming bug that the disabled-abort clear function which is to constitute the counterpart of the abort disable function in a pair, is absent.

Detailed Description Text (29):
The preempt disable processing and the disabled-preempt clearing processing can be carried out by making use of the disabled-preempt clearing interface and the disabled-preempt clearing interface disclosed in Japanese Patent Application No. 8-97997 mentioned hereinbefore. Namely, the following interfaces are employed.

Detailed Description Text (31):
preempt_disable(*level)

Detailed Description Text (33):
level: the depth of the nest formed by a pair of the instant function now of concern (i.e., "preempt_disable") and the function "preempt_enable" is returned.

Detailed Description Text (35):
The function now of concern makes the currently executed process transit to the preempt-disabled state. This function "preempt_disable" and the function "preempt_enable" may be issued in a pair during the section or interval intervening between the transition to the preempt-disabled state in response to the issuance of this function and the restoration to the preempt-enabled state in response to the issuance of the function "preempt_enable". This means that the pair of the function now of concern and the function "preempt_enable" can be nested. The function now of concern and the function "preempt_enable" issued internally of the nest make no state transition between the preempt-disabled state and the preempt-enabled state. The argument "level" reflects the depth of such nest.

Detailed Description Text (37):
preempt_enable(level)

Detailed Description Text (39):
level: the level of the nest returned by the function "preempt_disable" which is to constitute a counterpart of the pair of the instant function is designated.

Detailed Description Text (41):
The function of concern can make the currently executed process restore to the preempt-enabled state. The function "preempt_disable" and this function may be issued in a pair during the interval or period spanning over the transition to the preempt-disabled state in response to the issuance of the function "preempt_disable" and the restoration to the preempt-enabled state in response to the issuance of this function. This means that the pair of the function "preempt_disable" and the function of concern can be nested. The function "preempt_disable" and the function now of concern issued internally of the nest make no state transition between the preempt-disabled state and the preempt-enabled state. The argument "level" designates the depth of such nest, i.e., the value of "level" obtained due to the issuance of the counterpart function "preempt_disable". The depth of the nest is held by the operating system as well. Thus, discrepancy of this value with the depth of the nest specified by the argument, the so-called error return takes place.

Detailed Description Text (42):
When the function 40 included in the process 200 uses the preempt disable function 32, the function 40 stores in its own work area 37 the current depth of the nest returned from the preempt disable function 32. For issuing the disabled-preempt clear function 33, the value stored in the work area 23 is designated or specified as the argument of the function 40. The disabled-preempt clear function serves to compare the value mentioned above with the current depth of the nest to thereby validate the error return unless coincidence is found between both the values mentioned above. By virtue of this function, such a programming bug can easily be detected that the disabled-preempt clear function which is to constitute the counterpart of the preempt disable function in a pair is absent.

Detailed Description Text (53):
At this juncture, it should be mentioned that the serializer process 220 in the system shown in FIG. 12 operates constantly in the preempt-enabled state.

Detailed Description Text (55):
By adopting the process executing method based on the serializer process described above, the exclusive control (i.e., mutual exclusion control or management) need not be performed because the process which can perform processing directly on or for the shared resource is only one, i.e., the serializer process. Thus, by virtue of the arrangement that the processing for the shared resource is performed by the preempt-enabled independent process dedicated to the processing for the shared resource, process execution can be accomplished without exerting any influence to the periodical driving performance (or periodical drivability) of other processes partaking in the continuous media processing.

Detailed Description Text (60):
An resource identifier 600 shown in FIG. 17A is an argument of 64 bits contained in the system call issued by a user application for using the same in the access to the resource. Of the 64-bit argument, the leading 32 bits are used for identifying a starting or leading address 601 of a resource 608 while the trailing 32 bits constitute generation identifying information 602. The resource identifier 600 is generated upon creation of the resource and used for making access to the resource until the resource is released or deallocated.

Detailed Description Text (61):
A resource generation counter 603 shown in FIG. 17B is created for each of the processes upon generation thereof and serves as a counter for recording the number of times the resource is generated. The count value 604 of the resource generation counter 603 is set to the initial value of zero and incremented by one every time the resource is created.

Detailed Description Text (62):
Generation identifying information 605 shown in FIG. 17C contains leading 16 bits representing a count value 606 of the resource creation counter with the trailing 16 bits representing a process identifier 607.

Detailed Description Text (63):
Hereinafter, the resource which contains the generation identifying information 609 at a leading or starting location thereof will be referred to as the resource 608.

Detailed Description Text (66):
(1) In a step 2200, a user application issues a system call requesting a process generating processing to the operating system, which responds thereto by creating a resource generation counter 603 for the process as generated.

Detailed Description Text (67):
(2) In a step 2201, the user application issues to the operating system a system

call for the resource allocation processing.

Detailed Description Text (68):
(3) In a step 2202, the user application issues to the operating system a system call requesting an access processing to the resource allocated in the step 2201.

Detailed Description Text (69):
(4) In a step 2203, the user application issues to the operating system a system call requesting an deallocation processing of the resource allocated in the step 2201.

Detailed Description Text (70):
In the following, the processing for each of the system calls mentioned above will be described in detail.

Detailed Description Text (72):
FIG. 19 shows a flow chart for illustrating the resource allocation processing step 2201 shown in FIG. 18. The resource generation counter is incremented by one (step 2300 in FIG. 19), wherein generation identifying information creating or making processing "make_idinf" is executed (step 2301), which is then followed by step 2302 in which a resource area allocation (or memory area acquisition) processing "alloc_mem" is executed and step 2303 of creating an identifier "make_id" is executed. Thereafter, the generation identifying information is stored at leading 32 bits of the resource area as returned (step 2304).

Detailed Description Text (74):
FIG. 20 shows a flow chart for illustrating the generation identifying information creating processing in step 2301 shown in FIG. 19. Hereinafter, this processing will also be referred to as "make_idinf" processing. The generation identifying information is created by a pair of the resource generation counter value and the process identifier of the process destined for the resource generation. Referring to FIG. 20, the process identifier is first acquired (step 2400), wherein the resource generation counter value is placed in the leading 16 bits while a random number value as generated is stored in the trailing 16 bits, to thereby create or make the 32-bit generation identifying information (step 2401). The generation identifying information as created is returned (step 2402).

Detailed Description Text (76):
FIG. 21 shows a flow chart for illustrating the resource acquisition (or allocation) processing in the step 2302 shown in FIG. 19. This processing will also be referred to as "alloc_mem" processing. Because the resource area is constituted by a generation identifying information area and a resource area, the resource area size added with the generation identifying information data size of 32 bits is determined (step 2500), wherein the resource area of the size as determined is allocated (step 2501). Thereafter, the leading address of the resource area (memory area) is returned (step 2502).

Detailed Description Text (77):
.sctn.2.1.3. Identifier Creating (make_id) Processing

Detailed Description Text (78):
FIG. 22 shows a flow chart for illustrating the identifier creating processing in the step 2303 shown in FIG. 19. Hereinafter, this processing will also be referred to as "make_id" processing. The identifier is constituted by a pair of the resource address and the generation identifying information. The address of the resource acquired in resource area allocation or "alloc_mem" processing (step 2302) is placed at leading 32 bits while the generation identifying information created in the generation identifying information creating "make_id" processing (step 2301) is placed at the trailing 32 bits, to thereby create the identifier of 64 bits (step 2600). The identifier as created is returned as the argument for use in making

access to the resource (step 2601).

Detailed Description Text (80):
FIG. 23 shows a flow chart for illustrating a resource access processing corresponding to the step 2202 in FIG. 18. Referring to FIG. 23, the leading 32 bits of the identifier received as the argument of the system call issued by the user application are extracted to thereby define the resource address while the trailing 32 bits are extracted for defining the generation identifying information S1 (step 2700). Subsequently, the leading 32 bits are read out from the resource on the basis of the resource address to define the generation identifying information S2, i.e., "read_mem" information (step 2701), wherein the generation identifying information S1 and S2 as extracted are compared with each other (step 2702). Unless the comparison results in coincidence, the access to the resource is not performed, and an error message is returned (step 2703). On the other hand, when the above comparison results in coincidence, then the access processing to the resource is performed (step 2704).

Detailed Description Text (83):
Similarly to the resource access processing (step 2202 shown in FIG. 18), in the resource deallocation processing (step 2203 shown in FIG. 18), the leading 32 bits of the identifier received as the argument of the system call issued by the user application are extracted to define the resource address while the trailing 32 bits are extracted for defining the generation identifying information S1 (step 2800). Subsequently, the leading 32 bits are read out from the resource to define the generation identifying information S2 "read_mem" processing (step 2801). Then, the generation identifying information S1 and S2 mentioned above are compared with each other (step 2802). Unless the comparison results in coincidence, the resource is not deallocated and a corresponding error message is returned (step 2803). On the other hand, when the above comparison results in coincidence, then the resource deallocation processing is performed (step 2804).

Detailed Description Text (85):
(1) As is obvious from the description concerning the resource access processing (.sctn.2.2.), the address for accessing the resource is contained in the identifier which is the argument of the system call issued by the user application to the operating system. Thus, it is possible to make access to the resource by using the identifier in place of using the hash function for the translation between the identifier and the address. Consequently, overhead involved by the use of the hash function can be diminished.

Detailed Description Text (86):
(2) Let's suppose that a process A and a process B share a resource and that the process A has issued a system call for deallocating the resource X. In that case, in the conventional system, the operating system searches the identifiers contained in the resource management data by following the pointers, starting from the index derived by using the hash function to thereby determine the address of the resource to be deallocated. Further, other processes sharing the resource X is searched to release all the process management data as found and issue a message indicating that the resource X is invalid. In the meantime, the operating system inhibits or disables all the accesses to the shared resource from the user applications.

Detailed Description Text (87):
By contrast, according to the methods of the invention, the address of the resource can be obtained for unlocking the resource without need for following the pointers for acquiring the address of the resource because the address has already been contained in the identifier of the resource X to be deallocated. Additionally, it is apparent from the previous description concerning the resource deallocation processing (.sctn.2.3.), the operating system controls or manage the resource access right on the basis of only the result of comparison between the generation identifying information S1 contained in the identifier assigned to the resource X

and the generation identifying information S2 stored in the leading of the resource X. Thus, the task imposed on the operating system is only the deallocation of the resource X.

Detailed Description Text (88):
In this manner, the duration of the state in which the access to the resource is disabled and which makes appearance between the successive process completion processings can be reduced to only the time involved in deallocating the resource X.

Detailed Description Text (89):
(3) In conjunction with the resource deallocation processing described above, it is again assumed that a process C generates a resource Y and allocates the created resource Y to the address having been allocated to the resource X in the state in which the resource X is deallocated or released. In that case, the operating system issues to the process B no information or message indicating that the resource X has been deallocated. Consequently, there may arise such situation that the process B tries to make access to the resource Y on the basis of the address information contained in the identifier of the process B. However, because the identifier contains not only the address information but also the generation identifying information and because the generation identifying information is different between the identifier assigned to the resource X and the identifier assigned to the resource Y, the process B trying to make access to the resource Y with its own identifier can not access the resource Y. In this manner, the illegal access can be positively inhibited or disabled.

Detailed Description Text (94):
When a resource is created, the value of the system clock is fetched as the generation identifying information which is then stored at a leading address of the resource. An identifier of 64 bits is created which is composed of 32 MSB (more significant bits) corresponding to the address of the resource and 32 LSB (less significant bits) corresponding to the generation identifying information.

Detailed Description Text (95):
For making access to the resource, the generation identifying information is extracted from the identifier transferred as the argument of the system call from the user application to the operating system, wherein the extracted generation identifying information is compared with the generation identifying information stored at the leading or starting address of the resource. When this comparison results in coincidence, the access to the resource is enabled, i.e., allowed. On the contrary, when the comparison shows discrepancy, the access to the resource is disabled or inhibited with an error message being returned.

Detailed Description Text (98):
Upon creation of a resource, the above-mentioned generation identifying information is incremented by one and stored at the starting or leading address of the resource. Subsequently, a 64-bit identifier is created which includes more significant 32 bits corresponding to the address of the resource and less significant 32 bits corresponding to the generation identifying information.

Detailed Description Text (99):
For accessing the resource, the resource access processing is performed by using the identifier mentioned above similarly to the processing described in the section .sctn.3.1.

Detailed Description Text (100):
Further, the resource accessing methods described above by reference to FIGS. 17A to 23 allow to structure a safe system which inhibits or disables the invalid access to the shared resource by applying the accessing method to the processing steps 1302 and 1306 shown in FIG. 7. This shared resource accessing method will be

described below.

Detailed Description Text (101):
For the shared resource allocation, the processing illustrated in FIG. 19 is executed for effecting the shared resource allocation. At that time, the identifier assigned to the shared resource is held to check the validity of the shared resource by using this identifier upon access to the shared resource.

Detailed Description Text (103):
When the processings 1302 and 1306 shown in FIG. 7 are executed for carrying out the shared resource access processing, processing illustrated in FIG. 23 is executed for validating the access to the shared resource. At first, before making access to the shared resource, the generation identifying information (602 in FIG. 17A) stored in the identifier is compared with the generation identifying information (609 in FIG. 17A) of the shared resource (step 2702 in FIG. 23).

Detailed Description Text (104):
When coincidence is found between both the generation identifying information, it is then decided that the identifier of the shared resource is valid, i.e., the resource address (601 in FIG. 17A) contained in the identifier is valid. Accordingly, access is made to the shared resource by using the resource address to perform the processing for the shared resource.

Detailed Description Text (105):
On the contrary, when discrepancy is found between both the generation identifying information mentioned above, this means that the identifier of the shared resource is invalid. More specifically, assuming, by way of example, that discrepancy of the generation identifying information occurs in the processing step 1306 shown in FIG. 7, this means that the shared resource is deallocated for some reason during the preempt-enable interval intervening between the release of the preempt disabled-state (step 1303 in FIG. 7) in succession to the completion of the processing 1302 shown in FIG. 7 and the next preempt disabling (step 1305 in FIG. 7). Thus, the address of the resource (shared resource) as contained in the identifier is invalid. Consequently, the access to the shared resource is suspended and an error processing (step 2703 in FIG. 23) is carried out. In the error processing, the preempt disabling or inhibition and/or abort disabling may be cleared as occasion requires.

Detailed Description Text (106):
Owing to the processing method described above, deallocating of the shared resource carried out outside of the preempt-disabled interval can be detected. Besides, the access to the invalid resource which may occur in accompanying the above-mentioned shared resource deallocating can be prevented for thereby protecting the resource from being destroyed. In this way, it is possible to structure a system capable of processing the shared resource with enhanced security even when the preempt enable interval is set in the shared resource processing interval or section.

Detailed Description Text (107):
As will now be appreciated from the foregoing description, according to the teachings of the invention disclosed herein, overhead involved in the translation between the address and the identifier as well as search for checking the presence/absence of the access right can be reduced with the access performance being correspondingly enhanced. Besides, the resource-access disabled interval or section taking place upon completion of the process can be shortened. Additionally, the resource can be protected against destruction due to the illegal access to the resource from the process imparted with no resource access right.

CLAIMS:

1. A process executing method for executing a given one of a plurality of processes

h e b b g e e e f c e h e ge

in a computer system by using one shared resource which is accessed by said plurality of processes executed on a processor, said method comprising the steps of:

a) disabling abortion of said given one process;

b) disabling preemption of said given one process;

c) processing said shared resource for use by said given one process after disabling preemption, and enabling preemption after processing said shared resource;

d) disabling preemption before succeeding processing said shared resource and enabling preemption after processing said shared resource;

e) enabling abortion of said given one process after enabling preemption; and

f) after enabling abortion, executing a forcive termination request issued for said given one process during a period in which said given one process was in an abort-disabled state.

L20: Entry 4 of 6 File: USPT Dec 17, 1996

DOCUMENT-IDENTIFIER: US 5586289 A
TITLE: Method and apparatus for accessing local storage within a parallel processing computer

Abstract Text (1):
A processor within a parallel processing computer having a plurality of processors, where each processor is directly connected to a local storage memory. Each processor contains a principal processing element (PPE), a memory controller, and a multiplexor. The PPE executes a series of program instructions including local storage memory access instructions that cause the PPE to produce a local storage memory access request for accessing information within the local storage memory. The memory controller is connected to the PPE and a plurality of information resources of the parallel processing computer. This controller selectively routes local storage memory access requests from the information resources to an output port of the memory controller and generates an enable flag that is set to a first state when a selected one of the plurality of information resources can access the local storage memory and is set to a second state when the PPE is accessing the local storage such that access by the information resources is deferred until access by the PPE is complete. The multiplexor is connected to the PPE, the memory controller and the local storage memory. The multiplexor, operating in response to the enable flag, multiplexes the PPE access request with the selected information resource access requests such that access by the PPE to the local storage memory occurs substantially without time-delay interruption of the execution of program instructions by the PPE.

Brief Summary Text (5):
The problem of efficiently accessing dedicated local storage by a plurality of parallel processors has been addressed in certain prior art patents.

Brief Summary Text (6):
For example, in U.S. Pat. No. 4,837,676 a computer architecture is described which attempts to achieve highly parallel execution of programs in instruction flow form. In this patent, individual units, such as process control units, programmable function units, and memory units are individually coupled together by an interconnection network as self-contained units. Each process control unit initiates its assigned processors in sequence, routing instruction packets for each process through the computer network and an address network in order to provide time share of a single communications highway by multiple instruction packet streams.

Detailed Description Text (8):
In normal operation, the PPE 285 always has access to local memory 6. Local memory 6 is accessed by providing an address (PPE Addr) to MUX 240, and then either storing data via the PPE Data Out connection to MUX 260, or reading data from local memory via the PPE Data-In connection. The instant invention serves to create an access to local memory at any time that local memory is not being used by the PPE.

Detailed Description Text (16):
The LMMemEn output of MEM CTRL selects whether the memory access source is the PPE 255 or MEM CTRL 230. In addition, the LM Data-Out output permits data transfer into

memory 6, while the LM Wr En output enables MUX 250 for a write operation, and the LM Addr output allows an address to be applied to memory 6.

Detailed Description Text (17):
In FIG. 3 there is shown in greater detail the logic included within the MEM CTRL unit 230 of FIG. 2. The circuitry in FIG. 3 consists of Finite State Machine Logic (FSML) 301, Load Logic 302, Execution Unit 303, Gating Logic 304, Enable MUX 305, Data MUX 306, Address Masks 307, and MUX 308.

Detailed Description Text (20):
The primary output of the Load Logic is either address plus data for storage in local memory, or just the address that will be needed in order to read data from local memory.

Detailed Description Text (23):
Address Masks 307 and MUX 308, in conjunction with AND gate 304, serve to mask preselected bits from the data forwarded from the Load Logic to the Execution Unit. This function serves to separate the address from the data being forwarded to the Execution Unit.

Detailed Description Text (27):
Also applied to MUX 310 is a three-bit input from flag-polling counter 312. Counter 312 is driven at its clock input by the processor system clock (not shown), which provides clock timing signals for all of the processors. Counter 312 is adopted to cycle through the eight three-bit binary codes in a cyclical fashion in response to the processor clock signals when a signal of logic level "1" is applied to counter 312 from Active Register 314. A logic level "0" results in counter 312 maintaining its present value.

Detailed Description Text (28):
MUX 311 is a two-by-three MUX having one input port assigned a three-bit binary code "000". The other input port is connected to the output of counter 312.

Detailed Description Text (29):
Flip-flop 314 serves to enable counter 312 upon receipt of the next SRC signal which "sets" flip-flop 314.

Detailed Description Text (53):
In FIG. 5, the Execution Unit consists of a "B" unit 500, a "C" unit 505, a "D" unit 501, and an "E" unit 509. Also included is Conditional Operation Comparator 502, Opcode Register 504, Conditional Operation Register 503, Register 506, Counter 508, and Comparator 507.

Detailed Description Text (101):
6. f output: Increments Addr Counter for next memory access.

Detailed Description Text (106):
11. 1 output: Asserts LMMemEn, which specifies that HL is providing the address.

Detailed Description Text (114):
internal WRn counter reset to 0. (WRn value stored in 2 bit counter 632).

Detailed Description Text (124):
Assert F output signal (increments Addr Counter)

Detailed Description Text (130):
Write WRn counter value to h output, increment WRn Counter.

Detailed Description Text (133):
Assert f output signal (increments Addr Counter)

Detailed Description Text (137):
Write WRn counter value to h output, increment WRn Counter. (WRn value sent to Store Logic, and WRn Comparator.

Detailed Description Text (140):
Assert f output signal (increments Addr Counter).

Detailed Description Text (142):
Also included is a 2-bit counter 632. The counter 632 includes a Reset input port, an Incr input port, and a 2-bit-wide output port, which is the WRn output port.

Detailed Description Text (147):
As shown in FIG. 6, there is included a two-input OR gate 672, a three-input OR gate 674, and a three-input OR gate 675. The circuit also includes a MUX 676 and a delay unit 678. The MUX 676 is a 1-by-4 MUX having four control-signal input ports, a Resource Idle output port, and an Active SRC input port. The two inputs of the gate 672 are connected respectively to the output of the gate 646, and the output of the gate 648. The output of the gate 672 is connected to the increment port 636 of the counter 632.

Detailed Description Text (150):
The output of the delay unit 660 is connected to the CompEn output port. The Reset input port of the counter 632 is connected to the Start input port of the circuit in FIG. 6.

CLAIMS:

1. In a parallel processing computer having a plurality of processors, where each processor is directly connected to a local storage memory, each of said processors comprising:

a principal processing element (PPE) for executing a series of program instructions including local storage memory access instructions that cause the PPE to produce a local storage memory access request for accessing information within the local storage memory;

a memory controller, connected to said PPE and a plurality of information resources of the parallel processing computer, for selectively routing local storage memory access requests from said information resources to an output port of said memory controller and for generating an enable flag that is set to a first state when a selected one of said plurality of information resources can access said local storage memory and is set to a second state when said PPE is accessing said local storage, where generating said first state is deferred until said PPE has completed accessing said local storage memory; and

multiplexing means, connected to said PPE, said memory controller and said local storage memory, for multiplexing, in response to said enable flag, said PPE access request with said selected information resource access requests, where said first state of said enable flag causes said multiplexing means to permit said selected information resource to access said local storage memory and said second state of said enable flag causes said multiplexing means to permit said PPE to access said local storage memory substantially without time-delay interruption of the execution of program instructions by the PPE; wherein said memory controller further comprises:

a monitor circuit, connected to each of said information resources, for identifying the access requests generated by said information resources and for selecting one of said plurality of information resources to access said local storage memory and for generating, in response to said access request of said selected information

resource, a request signal identifying said selected information resource.

2. The processor of claim 1 wherein said memory controller further comprises:

an access circuit, connected to said monitor circuit, for transferring information between the local storage memory and said selected information resource, and where said access circuit generates an access-completed signal for signaling that the transfer of information between said local storage memory and said selected information resource has been completed.

3. The processor of claim 2 wherein said access circuit further comprises:

load logic, connected to said monitor circuit and said plurality of information resources, for generating a memory address for said local storage memory to access information from said local storage memory in response to said access request from said selected information resource, and, if said access request from said information resource is a write request, said load logic passes data from said selected information resource to a data output port of said load logic; and

an execution unit, connected to said load logic, for transferring said memory address to said multiplexing means and, if said access request from said selected information resource is a write request, said execution unit passes data from said load logic to said multiplexing means.

5. The processor of claim 4 wherein said double buffers are capable of storing variable length data words, where data words generated by a first information resource has a different length than a data word generated by a second information resource.

6. In a parallel processing computer having a plurality of processors, where each processor is directly connected to a local storage memory and each processor includes a principal processing element (PPE) for executing a series of program instructions including local storage memory access instructions that cause the PPE to produce a local storage memory access request for accessing information within the local storage memory, and a memory controller, connected to said PPE and a plurality of information resources of the parallel processing computer, a method for accessing said local storage memory comprising the steps of:

selectively routing local storage memory access requests from a selected one of said information resources to an output port of said memory controller;

generating an enable flag that is set to a first state when one of said plurality of resources can access said local storage memory and is set to a second state when said PPE is accessing said local storage, where generating said first state is deferred until said PPE has completed accessing said local storage memory; and

multiplexing, in response to said enable flag, said PPE access request with said selected information resource access requests, where said first state of said enable flag permits said selected information resource to access said local storage memory and said second state permits said PPE to access said local storage memory substantially without time-delay interruption of the execution of program instructions by the PPE; wherein the method further comprises the steps of:

identifying the access requests generated by said information resources;

selecting an information resource from said plurality of information resources for accessing said local storage memory; and

generating, in response to said access request from said selected information resource, a request signal identifying said selected information resource.

8. The method of claim 7 further comprising the steps of:

generating a memory address for said local storage memory to access information from said local storage memory in response to said access request from said selected information resource; and

transferring said memory address to said multiplexor and, if said access request is a write request, passing data to said multiplexor.

10. The method of claim 9 wherein said double buffer is capable of storing variable length data words, where data words generated by a first information resource has a different length than a data word generated by a second information resource.

L18: Entry 9 of 22 File: USPT Oct 5, 1999

DOCUMENT-IDENTIFIER: US 5963142 A
TITLE: Security control for personal computer

Abstract Text (1):
A personal computer provides security features enabling control over access to data retained in the computer. The computer is secured by having the system ROM provide a password at power-on to a security device which controls access to the secured features. Once a password has been downloaded to the security device, a Protect Resources command is issued to the security device. To gain access to the secured feature after boot-up, the user provides the correct password to the security device and waits for approval from the security device. Since the security device only verifies the password and does not divulge it, security of the system is enhanced. Once access to protected resources is no longer required, the computer issues another Protect Resources command to the security device to once more lock access to the protected resources.

Detailed Description Text (3):
Referring now to FIG. 1, a computer system S according to the present invention is shown. In the preferred embodiment, there are two primary buses located in the system S. The first bus is the PCI or Peripheral Component Interconnect bus P which includes an address/data portion and control signal portion. The second primary bus in the system S is the ISA bus I. The ISA bus I includes an address portion, a data portion 110, and a control signal portion 112. The PCI and ISA buses P and I form the backbones of the system S.

Detailed Description Text (4):
A CPU/memory subsystem 100 is connected to the PCI bus P. The processor 200 is preferably the Pentium processor from Intel, preferably operating externally at 50 or 60 MHz, but could be an 80486 from Intel or processors compatible with the 80486 or Pentium or other processors if desired. The processor 200 provides data, address, and control portions 202, 204, 206 to form a host bus HB. A level 2 (L2) or external cache memory system 208 is connected to the host bus HB to provide additional caching capabilities to improve performance of the computer system. The L2 cache 208 may be permanently installed or may be removable if desired. A cache and memory controller and PCI bridge chip 210, such as the 82434X chip from Intel Corporation or the chip described in patent applications Ser. Nos. 08/324,016, entitled "SINGLE BANK, MULTIPLE WAY CACHE MEMORY" pending and 08/324,246, entitled "SYSTEM HAVING A PLURALITY OF POSTING QUEUES ASSOCIATED WITH DIFFERENT TYPES OF WRITE OPERATIONS FOR SELECTIVELY CHECKING ONE QUEUE BASED UPON TYPE OR READ OPERATION", filed Oct. 14, 1994, now U.S. Pat. No. 5,634,073, and hereby incorporated by reference, is connected to the control portion 206 and to the address portion 204. The bridge chip 210 is connected to the L2 cache 208 as it incorporates the cache controller and therefore controls the operation of the cache memory devices in the L2 cache 208. The bridge chip 210 is also connected to control a series of data buffers 212. The data buffers 212 are preferably similar to the 82433LX from Intel, or those described in patent applications Ser. Nos. 08/324,246 now U.S. Pat. 5,634,073 as incorporated above and 08/323,263 entitled "DATA ERROR DETECTION AND CORRECTION SYSTEM", filed Oct. 14, 1994, now U.S. Pat. No. 5,555,250 and hereby incorporated by reference, and are utilized to handle memory data to a main memory array 214. The data buffers 212 are connected to the

processor data portion 202 and receive control signals from the bridge chip 210. The data buffers 212 are also connected to the PCI bus P for data transfer over that bus. The data buffers 212 provide a memory data bus 218 to the memory array 214, while a memory address and memory control signal bus 220 is provided from the bridge chip 210.

Detailed Description Text (7):
A PCI-ISA bridge 130 is provided to convert signals between the PCI bus P and the ISA bus I. The PCI-ISA bridge 130 includes the necessary address and data buffers and latches, arbitration and bus master control logic for the PCI bus, ISA arbitration circuitry, an ISA bus controller as conventionally used in ISA systems, an IDE (integrated drive electronics) interface, and a DMA controller. Preferably the PCI-ISA bridge 130 is a single integrated circuit, but other combinations are possible. A series of ISA slots 134 are connected to the ISA bus I to receive ISA adapter cards. A series of IDE slots 133 are connected to the ISA bus I and the PCI-ISA bridge chip 130 to receive various IDE devices, such as hard disk drives, tape drives and CD-ROM drives. A series of PCI slots 135 are connected to the PCI bus P to receive PCI adapter cards.

Detailed Description Text (8):
A combination I/O chip 136 is connected to the ISA bus I. The combination I/O chip 136 preferably includes a floppy disk controller, real time clock (RTC), CMOS memory, two UARTs, various address decode logic and security logic to control access to the CMOS memory and the power on password values. A floppy disk connector 138 for receiving a cable to a floppy disk drive is connected to the combination I/O chip 136 and the ISA bus I. Serial port connectors 137 are also connected to the combination I/O chip 136. A buffer 144 is connected to the ISA bus I to provide an additional X bus X for various additional components of the computer system. A flash ROM 154 receives its control, address and data signals from the X bus X. Preferably the flash ROM 154 contains the BIOS information for the computer system and can be reprogrammed to allow for revisions of the BIOS. An 8042 or keyboard controller 156 is connected to the X bus X and ISA bus I address and control portion. The keyboard controller 156 is of conventional design and is connected in turn to a keyboard connector 158 and a mouse or pointing device connector 160.

Detailed Description Text (9):
A miscellaneous system logic chip 132 is connected to the X bus X. The miscellaneous system logic chip 132 contains counters and timers as conventionally present in personal computer systems, an interrupt controller for both the PCI and ISA buses P and I, enhanced parallel port circuitry and power management logic, as well as other miscellaneous circuitry. Additionally, the miscellaneous system logic chip 132 includes circuitry of a security management system according to the present invention and so is connected to the flash ROM 154 through write protection logic 540.

Detailed Description Text (12):
Commands are preferably issued from the computer to the security device at a predetermined address. Status may be read from the last resource, or slot, indexed. The security device is capable of protecting a plurality of resources or slots. In the preferred embodiment shown in detail, only a single resource is protected, but at various locations reference is explicitly made to the plurality of resources or slots and those skilled in the art can readily determine appropriate modifications where not specifically discussed. A Read Status command is configured so that the status register of any particular resource, or slot, can be read, without affecting other operations occurring to a resource or slot, such as unlocking or changing a password. Access to a specific resource, or slot, in the security device is performed with an indexed address scheme. Two addresses are used for the indexing scheme, one address is for commands, while the second address is the data/status register. The second address acts as a data register for a write cycle and as a status register during a read cycle. The index is placed in the upper three bits of

the command register.

Detailed Description Text (16):
In addition to sending commands to the security device, status can be read from the security device at any time, by reading the status/data register, preferably located at a second predetermined address adjacent to the first predetermined address. The data/status register serves two purposes. When serving as the data register, the register is used when storing passwords or verifying passwords (accessing resources).

Detailed Description Text (19):
Bits 7-5 provide an indication of which particular resource or slot the remaining bits identify. The PL bit indicates when set that this resource is permanently locked. The D bit indicates when set that a one second delay is in progress due to a password mismatch. The U bit is the state of the UNLOCK.sub.-- pin or signal for the resource.

Detailed Description Text (28):
For simplicity, while this description focuses on protecting only one resource, it is within the scope of the invention to have multiple resources located in multiple slots for passwords in the security device. This is done by the use of the different index values, each referencing a different resource or slot. In this description, the index value is always assumed to be zero to access the first slot. Thus, in a multiple resource embodiment, there are a plurality of eight byte password registers used to store passwords. In the multiple resource embodiment, the security device has an UNLOCK_ output for each slot. The additional slots in the security device can include slots for power-on password, administrator password, Safe Start hash codes, among others. The power-on password slot controls the power-on password, which is currently only changeable at boot time. If a slot is provided then the power-on password can be changed at run-time. In addition to the power-on password slot, another slot may be used to address the rest of the protected areas that are accessible through the Administrator password. Further, Safe Start codes resource is another candidate.

Detailed Description Text (32):
The transition from the ACC.sub.-- RESOURCE state 442 to the IDLE state 440 occurs when data is written to the data/status register (WRITE.sub.-- DATA); the password count equals zero (PASSWD.sub.-- CNT=0), indicating that all eight bytes have just been written; and the SET.sub.-- MISMATCH and MISMATCH signals are deasserted. The SET.sub.-- MISMATCH signal is set when the current byte being written is a mismatch, while the MISMATCH signal indicates that a byte previously written on this attempt mismatched. The transition from the ACC.sub.-- RESOURCE state 442 to the IDLE state 440 also occurs when a command other than a Read Status command is written to the command register. The transition from the ACC.sub.-- RESOURCE state 442 to the DELAY state 446 occurs when data is written to the data/status register (WRITE.sub.-- DATA); the password count equals zero, indicating the last byte has been written; and either the SET.sub.-- MISMATCH or the MISMATCH signal is asserted. This transition handles the event where the key and the password do not match. A delay of preferably one second is encountered once the DELAY state 446 is entered. This period is clocked by a count-down counter 536 (FIG. 5) whose output is DELAY.sub.-- COUNT. Thus, upon DELAY.sub.-- COUNT reaching zero, the DELAY state 446 transitions back to the IDLE state 440.

Detailed Description Text (37):
When the state machine 438 is in state ACC.sub.-- RESOURCE 442 and data is written to the data register, the EN.sub.-- PASSWD.sub.-- CNT signal is asserted via an AND gate 474 and the OR gate 470. Further, the output of the AND gate 474 is ANDed via AND gate 476 with the output of a comparator 482 which compares the proper byte in a password register 478 with a key register 480 to generate the SET.sub.-- MISMATCH signal. The password register 478 is actually the depth of the password, eight

bytes in the preferred embodiment, and the proper byte is selected based on the password counter 528 value. The key register 480 need only be a single byte register as the previous bytes of the key need not be stored. The SET.sub.-- MISMATCH signal indicates a mismatch of the current byte and is latched to indicate that a mismatch has already occurred in a previous clock period. The output of the latching operation, the MISMATCH signal, is gated with the SET.sub.-- MISMATCH signal using a NOR gate 484. The output of the NOR gate 484 is ANDed with the output of the AND gate 474 and the PASSWD.sub.-- CNT=0 signal by an AND gate 486 to generate the CLR.sub.-- PROT signal. The output of the NOR gate 484 is further inverted by an inverter 490 and then gated with the output of the AND gate 474 and the PASSWD.sub.-- CNT=0 signal by an AND gate 492 to generate the delay signals SET.sub.-- DLY and SET.sub.-- DLY.sub.-- CNT.

Detailed Description Text (41):
Two counters 528 and 536 are used to sequence the password count and the delay count, respectively. Both counters 528 and 536 have the data inputs connected to logic high, or 5V DC, and the COUNT UP/DOWN.sub.-- input wired to ground to indicate that the counters 528 and 536 are to count down. The SET.sub.-- PASSWD.sub.-- CNT signal is inverted by an inverter 524 and then ANDed with the RESET.sub.-- signal via an AND gate 526, which drives the inverted load or LD.sub.-- signal of the counter 528 to reload the counter 528. Similarly, the EN.sub.-- DLY.sub.-- CNT signal is inverted by an inverter 532 and then ANDed with the RESET.sub.-- signal via an AND gate 534 which drives the LD.sub.-- input of the counter 536. The detection that the output of each of counters 528 and 536 equals zero is performed by an OR gate. Thus, an OR gate 530 is connected to the outputs of counter 528 to generate the PASSWD.sub.-- CNT=0 signal. Similarly, an OR gate 538 is connected to the outputs of counter 536 to generate the DLY.sub.-- CNT=0 signal.

Detailed Description Text (42):
Turning to FIG. 6, the generation of the FRWP.sub.-- signal in the preferred embodiment is disclosed. This signal is generated in conjunction with the UNLOCK.sub.-- output of the security device, which is controlled through the states of the security device state machine as described above. In FIG. 6, the UNLOCK.sub.-- signal and bit 7 of the data portion of the X-bus are provided to OR gate 537. The output of the OR gate 537 is provided to the D input of a flip-flop 539. The CLR input of the flip-flop 539 is connected to the RESET.sub.-- signal for resetting purposes. Further, the flip-flop 539 is clocked by the falling edge of a write protect register address decode signal. A write protect register was provided in the combination I/O chip 136 and was used to enable or disable protection of the flash ROM in prior systems. The use of the bit in the register is maintained in the preferred embodiment and is supplemented by the use of the security device to allow further protection of the flash ROM. To guarantee that the flash ROM is properly protected, that register is mirrored in the miscellaneous system logic chip 132. The write protect register address decode signal indicates a write to that register. The output of the flip-flop 439 is an FRWP.sub.-- signal is used to control updating of the flash ROM. Thus to write to the flash ROM, the flash ROM resource must be unlocked using the security device and the bit in the write protect register must be set.

Detailed Description Text (44):
In addition to the connection from the write protection logic 540, the flash ROM 164 has address inputs which are coupled to the address portion of the X bus X, data signals which are coupled to the data portion of the X bus X, and conventional chip select, output enable, and write enable inputs that are driven by circuitry on the computer system S when the flash ROM 164 is addressed. Once the FRWP.sub.-- signal is true, the flash ROM 164 can be written in a manner similar to a random access memory (RAM). Thus, the updating of the flash ROM 164 can be accomplished in a secure manner.

L18: Entry 4 of 22 File: USPT Nov 12, 2002

DOCUMENT-IDENTIFIER: US 6480097 B1
TITLE: Security control for personal computer

Abstract Text (1):
A personal computer provides security features enabling control over access to data retained in the computer. The computer is secured by having the system ROM provide a password at power-on to a security device which controls access to the secured features. Once a password has been downloaded to the security device, a Protect Resources command is issued to the security device. To gain access to the secured feature after boot-up, the user provides the correct password to the security device and waits for approval from the security device. Since the security device only verifies the password and does not divulge it, security of the system is enhanced. Once access to protected resources is no longer required, the computer issues another Protect Resources command to the security device to once more lock access to the protected resources.

Detailed Description Text (3):
Referring now to FIG. 1, a computer system S according to the present invention is shown. In the preferred embodiment, there are two primary buses located in the system S. The first bus is the PCI or Peripheral Component Interconnect bus P which includes an address/data portion and control signal portion. The second primary bus in the system S is the ISA bus I. The ISA bus I includes an address portion, a data portion 110, and a control signal portion 112. The PCI and ISA buses P and I form the backbones of the system S.

Detailed Description Text (4):
A CPU/memory subsystem 100 is connected to the PCI bus P. The processor 200 is preferably the Pentium processor from Intel, preferably operating externally at 50 or 60 MHz, but could be an 80486 from Intel or processors compatible with the 80486 or Pentium or other processors if desired. The processor 200 provides data, address, and control portions 202, 204, 206 to form a host bus HB. A level 2 (L2) or external cache memory system 208 is connected to the host bus HB to provide additional caching capabilities to improve performance of the computer system. The L2 cache 208 may be permanently installed or may be removable if desired. A cache and memory controller and PCI bridge chip 210, such as the 82434X chip from Intel Corporation or the chip described in patent applications Ser. No. 08/324,016, entitled "SINGLE BANK, MULTIPLE WAY CACHE MEMORY" and Ser. No. 08/324,246, entitled "MEMORY CONTROLLER WITH WRITE POSTING QUEUES FOR PROCESSOR AND I/O BUS OPERATIONS AND ORDERING LOGIC FOR CONTROLLING THE QUEUES", filed Oct. 14, 1994, and hereby incorporated by reference, is connected to the control portion 206 and to the address portion 204. The bridge chip 210 is connected to the L2 cache 208 as it incorporates the cache controller and therefore controls the operation of the cache memory devices in the L2 cache 208. The bridge chip 210 is also connected to control a series of data buffers 212. The data buffers 212 are preferably similar to the 82433LX from Intel, or those described in patent applications Ser. No. 08/324,246 as incorporated above and Ser. No. 08/323,263 entitled "DATA ERROR DETECTION AND CORRECTION SYSTEM", filed Oct. 14, 1994, and hereby incorporated by reference, and are utilized to handle memory data to a main memory array 214. The data buffers 212 are connected to the processor data portion 202 and receive control signals from the bridge chip 210. The data buffers 212 are also connected

to the PCI bus P for data transfer over that bus. The data buffers 212 provide a memory data bus 218 to the memory array 214, while a memory address and memory control signal bus 220 is provided from the bridge chip 210.

Detailed Description Text (7):
A PCI-ISA bridge 130 is provided to convert signals between the PCI bus P and the ISA bus I. The PCI-ISA bridge 130 includes the necessary address and data buffers and latches, arbitration and bus master control logic for the PCI bus, ISA arbitration circuitry, an ISA bus controller as conventionally used in ISA systems, an IDE (integrated drive electronics) interface, and a DMA controller. Preferably the PCI-ISA bridge 130 is a single integrated circuit, but other combinations are possible. A series of ISA slots 134 are connected to the ISA bus I to receive ISA adapter cards. A series of IDE slots 133 are connected to the ISA bus I and the PCI-ISA bridge chip 130 to receive various IDE devices, such as hard disk drives, tape drives and CD-ROM drives. A series of PCI slots 135 are connected to the PCI bus P to receive PCI adapter cards.

Detailed Description Text (8):
A combination I/O chip 136 is connected to the ISA bus I. The combination I/O chip 136 preferably includes a floppy disk controller, real time clock (RTC), CMOS memory, two UARTs, various address decode logic and security logic to control access to the CMOS memory and the power on password values. A floppy disk connector 138 for receiving a cable to a floppy disk drive is connected to the combination I/O chip 136 and the ISA bus I. Serial port connectors 137 are also connected to the combination I/O chip 136. A buffer 144 is connected to the ISA bus I to provide an additional X bus X for various additional components of the computer system. A flash ROM 154 receives its control, address and data signals from the X bus X. Preferably the flash ROM 154 contains the BIOS information for the computer system and can be reprogrammed to allow for revisions of the BIOS. An 8042 or keyboard controller 156 is connected to the X bus X and ISA bus I address and control portion. The keyboard controller 156 is of conventional design and is connected in turn to a keyboard connector 158 and a mouse or pointing device connector 160.

Detailed Description Text (9):
A miscellaneous system logic chip 132 is connected to the X bus X. The miscellaneous system logic chip 132 contains counters and timers as conventionally present in personal computer systems, an interrupt controller for both the PCI and ISA buses P and I, enhanced parallel port circuitry and power management logic, as well as other miscellaneous circuitry. Additionally, the miscellaneous system logic chip 132 includes circuitry of a security management system according to the present invention and so is connected to the flash ROM 154 through write protection logic 540.

Detailed Description Text (12):
Commands are preferably issued from the computer to the security device at a predetermined address. Status may be read from the last resource, or slot, indexed. The security device is capable of protecting a plurality of resources or slots. In the preferred embodiment shown in detail, only a single resource is protected, but at various locations reference is explicitly made to the plurality of resources or slots and those skilled in the art can readily determine appropriate modifications where riot specifically discussed. A Read Status command is configured so that the status register of any particular resource, or slot, can be read, without affecting other operations occurring to a resource or slot, such as unlocking or changing a password. Access to a specific resource, or slot, in the security device is performed with an indexed address scheme. Two addresses are used for the indexing scheme, one address is for commands, while the second address is the data/status register. The second address acts as a data register for a write cycle and as a status register during a read cycle. The index is placed in the upper three bits of the command register.

Detailed Description Text (16):
In addition to sending commands to the security device, status can be read from the security device at any time, by reading the status/data register, preferably located at a second predetermined address adjacent to the first predetermined address. The data/status register serves two purposes. When serving as the data register, the register is used when storing passwords or verifying passwords (accessing resources).

Detailed Description Text (19):
Bits 7-5 provide an indication of which particular resource or slot the remaining bits identify. The PL bit indicates when set that this resource is permanently locked. The D bit indicates when set that a one second delay is in progress due to a password mismatch. The U bit is the state of the UNLOCK_ pin or signal for the resource.

Detailed Description Text (28):
For simplicity, while this description focuses on protecting only one resource, it is within the scope of the invention to have multiple resources located in multiple slots for passwords in the security device. This is done by the use of the different index values, each referencing a different resource or slot. In this description, the index value is always assumed to be zero to access the first slot. Thus, in a multiple resource embodiment, there are a plurality of eight byte password registers used to store passwords. In the multiple resource embodiment, the security device has an UNLOCK_ output for each slot. The additional slots in the security device can include slots for power-on password, administrator password, Safe Start hash codes, among others. The power-on password slot controls the power-on password, which is currently only changeable at boot time. If a slot is provided then the power-on password can be changed at run-time. In addition to the power-on password slot, another slot may be used to address the rest of the protected areas that are accessible through the Administrator password. Further, Safe Start codes resource is another candidate.

Detailed Description Text (32):
The transition from the ACC_RESOURCE state 442 to the IDLE state 440 occurs when data is written to the data/status register (WRITE_DATA); the password count equals zero (PASSWD_CNT=0), indicating that all eight bytes have just been written; and the SET_MISMATCH and MISMATCH signals are deasserted. The SET_MISMATCH signal is set when the current byte being written is a mismatch, while the MISMATCH signal indicates that a byte previously written on this attempt mismatched. The transition from the ACC_RESOURCE state 442 to the IDLE state 440 also occurs when a command other than a Read Status command is written to the command register. The transition from the ACC_RESOURCE state 442 to the DELAY state 446 occurs when data is written to the data/status register (WRITE_DATA); the password count equals zero, indicating the last byte has been written; and either the SET_MISMATCH or the MISMATCH signal is asserted. This transition handles the event where the key and the password do not match. A delay of preferably one second is encountered once the DELAY state 446 is entered. This period is clocked by a count-down counter 536 (FIG. 5) whose output is DELAY_COUNT. Thus, upon DELAY_COUNT reaching zero, the DELAY state 446 transitions back to the IDLE state 440.

Detailed Description Text (37):
When the state machine 438 is in state ACC_RESOURCE 442 and data is written to the data register, the EN_PASSWD_CNT signal is asserted via an AND gate 474 and the OR gate 470. Further, the output of the AND gate 474 is ANDed via AND gate 476 with the output of a comparator 482 which compares the proper byte in a password register 478 with a key register 480 to generate the SET_MISMATCH signal. The password register 478 is actually the depth of the password, eight bytes in the preferred embodiment, and the proper byte is selected based on the password counter 528 value. The key register 480 need only be a single byte register as the previous bytes of the key need not be stored. The SET_MISMATCH signal indicates a mismatch

of the current byte and is latched to indicate that a mismatch has already occurred in a previous clock period. The output of the latching operation, the MISMATCH signal, is gated with the SET_MISMATCH signal using a NOR gate 484. The output of the NOR gate 484 is ANDed with the output of the AND gate 474 and the PASSWD_CNT=0 signal by an AND gate 486 to generate the CLR_PROT signal. The output of the NOR gate 484 is further inverted by an inverter 490 and then gated with the output of the AND gate 474 and the PASSWD_CNT=0 signal by an AND gate 492 to generate the delay signals SET_DLY and SET_DLY_CNT.

Detailed Description Text (41):
Two counters 528 and 536 are used to sequence the password count and the delay count, respectively. Both counters 528 and 536 have the data inputs connected to logic high, or 5V DC, and the COUNT UP/DOWN_ input wired to ground to indicate that the counters 528 and 536 are to count down. The SET_PASSWD_CNT signal is inverted by an inverter 524 and then ANDed with the RESET_ signal via an AND gate 526, which drives the inverted load or LD_ signal of the counter 528 to reload the counter 528. Similarly, the EN_DLY_CNT signal is inverted by an inverter 532 and then ANDed with the RESET_ signal via an AND gate 534 which drives the LD_ input of the counter 536. The detection that the output of each of counters 528 and 536 equals zero is performed by an OR gate. Thus, an OR gate 530 is connected to the outputs of counter 528 to generate the PASSWD_CNT=0 signal. Similarly, an OR gate 538 is connected to the outputs of counter 536 to generate the DLY_CNT=0 signal.

Detailed Description Text (42):
Turning to FIG. 6, the generation of the FRWP_ signal in the preferred embodiment is disclosed. This signal is generated in conjunction with the UNLOCK_ output of the security device, which is controlled through the states of the security device state machine as described above. In FIG. 6, the UNLOCK_ signal and bit 7 of the data portion of the X-bus are provided to OR gate 537. The output of the OR gate 537 is provided to the D input of a flip-flop 539. The CLR input of the flip-flop 539 is connected to the RESET_ signal for resetting purposes. Further, the flip-flop 539 is clocked by the falling edge of a write protect register address decode signal. A write protect register was provided in the combination I/O chip 136 and was used to enable or disable protection of the flash ROM in prior systems. The use of the bit in the register is maintained in the preferred embodiment and is supplemented by the use of the security device to allow further protection of the flash ROM. To guarantee that the flash ROM is properly protected, that register is mirrored in the miscellaneous system logic chip 132. The write protect register address decode signal indicates a write to that register. The output of the flip-flop 439 is an FRWP_ signal is used to control updating of the flash ROM. Thus to write to the flash ROM, the flash ROM resource must be unlocked using the security device and the bit in the write protect register must be set.

Detailed Description Text (44):
In addition to the connection from the write protection logic 540, the flash ROM 164 has address inputs which are coupled to the address portion of the X bus X, data signals which are coupled to the data portion of the X bus X, and conventional chip select, output enable, and write enable inputs that are driven by circuitry on the computer system S when the flash ROM 164 is addressed. Once the FRWP_ signal is true, the flash ROM 164 can be written in a manner similar to a random access memory (RAM). Thus, the updating of the flash ROM 164 can be accomplished in a secure manner.

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 22 of 22 returned.

☐ 1. Document ID: US 6690402 B1

L18: Entry 1 of 22 File: USPT Feb 10, 2004

US-PAT-NO: 6690402
DOCUMENT-IDENTIFIER: US 6690402 B1

TITLE: Method of interfacing with virtual objects on a map including items with machine-readable tags

DATE-ISSUED: February 10, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Waller; Michael	London			GB
Mackay; Robin	London			GB
Ward; Matthew	London			GB

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
NCR Corporation	Dayton	OH			02

APPL-NO: 09/ 666656 [PALM]
DATE FILED: September 20, 2000

FOREIGN-APPL-PRIORITY-DATA:

COUNTRY	APPL-NO	APPL-DATE
GB	9922211	September 20, 1999
GB	0006161	March 14, 2000

INT-CL: [07] G09 G 5/00

US-CL-ISSUED: 345/850; 345/848, 345/852, 345/745
US-CL-CURRENT: 345/850; 345/745, 345/848, 345/852

FIELD-OF-SEARCH: 345/742, 345/744-747, 345/757, 345/848, 345/850, 345/851, 345/855, 345/864, 345/173, 340/995, 340/990

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

h e b b g e e e f e e ef b e

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5159180	October 1992	Feiler	
5736975	April 1998	Lunetta	
5878421	March 1999	Ferrel et al.	
5905251	May 1999	Knowles	
5945985	August 1999	Babin et al.	715/500.1
6014137	January 2000	Burns	345/747
6075502	June 2000	McDowall et al.	345/7
6091956	July 2000	Hollenberg	455/456
6119944	September 2000	Mulla et al.	235/472.03
6195093	February 2001	Nelson et al.	345/732
6204764	March 2001	Maloney	
6388688	May 2002	Schileru-Key	345/854
6414672	July 2002	Rekimoto et al.	345/173
6545660	April 2003	Shen et al.	345/156
6573916	June 2003	Grossweiler et al.	345/850

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
19743386	February 1993	DE	
0626635	November 1994	EP	
0 837 406	April 1998	EP	
WO 98/03923	January 1998	WO	
WO 98/38761	September 1998	WO	
WO 98/38762	September 1998	WO	
WO 98/49813	November 1998	WO	
WO 98/51036	November 1998	WO	
98/06055	December 1998	WO	
WO 00/45302	August 2000	WO	
0060440	October 2000	WO	

ART-UNIT: 2173

PRIMARY-EXAMINER: Cabeca; John

ASSISTANT-EXAMINER: Becker; Shawn

ATTY-AGENT-FIRM: Welte; Gregory A.

ABSTRACT:

Apparatus for accessing a displayable information resource and tailoring retrieved information to a user's requirements comprises a tag reader (18), a decoder (34) for identifying a coded resource address (38) carried by a tag (28), and access means for accessing the identified resource. A display means (14, 22, 24, 48) displays information loaded from the accessed information resource. The disclosed embodiment employs printed RF tag technology, and the information resource is an

h e b b g e e e f e e ef b e

Internet, intranet or extranet resource whose address (38) is a URL. The disclosed apparatus is embodied in an item of furniture, more specifically a table (10, 52). The tag reader (18) reads a tag (28) when a tagged item (26) is placed onto a support surface (14), and a display is presented by the support surface (14). The tagged item can, for example, be a product or its packaging, a ticket or token, or a letter or information sheet.

10 Claims, 23 Drawing figures

☐ 2. Document ID: US 6560698 B1

L18: Entry 2 of 22 File: USPT May 6, 2003

US-PAT-NO: 6560698
DOCUMENT-IDENTIFIER: US 6560698 B1

TITLE: Register change summary resource

DATE-ISSUED: May 6, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Mann; Daniel P.	Austin	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Advanced Micro Devices, Inc.	Sunnyvale	CA			02

APPL-NO: 09/ 306879 [PALM]
DATE FILED: May 7, 1999

INT-CL: [07] G06 F 9/44

US-CL-ISSUED: 712/248; 712/5, 712/224, 712/229, 712/247
US-CL-CURRENT: 712/248; 712/224, 712/229, 712/247, 712/5

FIELD-OF-SEARCH: 709/102, 712/227, 712/248, 712/5, 712/224-229, 712/247, 713/323, 713/300, 710/260, 710/62, 716/10, 717/4, 714/45, 714/46

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5684997	November 1997	Kau et al.	710/260
5727221	March 1998	Walsh et al.	713/310
5784291	July 1998	Chen et al.	716/10
5867717	February 1999	Milhaupt et al.	713/323
6192463	February 2001	Mitra et al.	712/43

h e b b g e e e f e e ef b e

6363442 March 2002 Hunter et al. 710/62

OTHER PUBLICATIONS

Intel Architecture Software Developer's Manual, vol. 3: System Programming Guide, Appendix A-Performance Monitoring Events, Intel Corp. (1997) 23 pages.
Intel Performance Evaluation & Analysis Kit--1394 Toolkit, Intel Corp., 2 pages, http://developer.intel.com/design/ipeak/1394.
Intel Performance Evaluation & Analysis Kit--ID Monitor: A WDM IO Monitoring Tool, Intel Corp., 2 pages, http://developer.intel.com/design/ipeak/iomon.

ART-UNIT: 2122

PRIMARY-EXAMINER: Khatri; Anil

ASSISTANT-EXAMINER: Kendall; Chuck

ATTY-AGENT-FIRM: Akin Gump Strauss Hauer & Feld LLP

ABSTRACT:

A microcontroller provides a register change summary resource for summarizing register changes. Selected system registers within each resource are coupled to bits in resource change registers of the register change summary resource using logic that tracks accesses to the system registers. Each resource change register is coupled to a bit in a summary register. For systems with numerous system registers, each summary register may be coupled to a bit in a higher-level summary register. The register change summary resource further provides a software-controlled bit mask register. A change in a summary or resource change register may trigger a processor interrupt. Each register in the register change summary resource can be reset, also under software control. The registers within the register change summary resource are accessible through a dedicated software development port.

25 Claims, 5 Drawing figures

☐ 3. Document ID: US 6532505 B1

L18: Entry 3 of 22 File: USPT Mar 11, 2003

US-PAT-NO: 6532505
DOCUMENT-IDENTIFIER: US 6532505 B1

TITLE: Universal resource access controller

DATE-ISSUED: March 11, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Stracovsky; Henry	San Jose	CA		
Szabelski; Piotr	Santa Clara	CA		

h e b b g e e e f e e ef b e

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Infineon Technologies AG	Munich			DE	03

APPL-NO: 09/ 439544 [PALM]
DATE FILED: November 12, 1999

INT-CL: [07] G06 F 13/12

US-CL-ISSUED: 710/63; 710/5, 710/6, 710/15, 710/17, 710/18, 710/19, 710/40, 710/62, 710/240, 710/244, 711/100, 711/147 , 711/150
US-CL-CURRENT: 710/63; 710/15, 710/17, 710/18, 710/19, 710/240, 710/244, 710/40, 710/5, 710/6, 710/62, 711/100, 711/147 , 711/150

FIELD-OF-SEARCH: 710/5, 710/6, 710/8, 710/15-20, 710/33, 710/36, 710/40, 710/62, 710/63, 710/64, 710/72, 710/74, 710/104, 710/107, 710/240, 710/244, 711/1, 711/2, 711/100, 711/147, 711/148, 711/150-152, 711/157, 711/158, 711/170, 711/200, 711/202, 711/211

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4354225	October 1982	Frieder et al.	364/200
4803623	February 1989	Klashka et al.	364/200
4831523	May 1989	Lewis et al.	364/200
5530845	June 1996	Hiatt et al.	395/500.48
5784582	July 1998	Hughes	710/117
5787457	July 1998	Miller et al.	711/105
5878240	March 1999	Tomko	395/311
5887199	March 1999	Ofer et al.	710/65
5923897	July 1999	Lipe et al.	710/5
5960212	September 1999	Mak	712/34
5987574	November 1999	Paluch	711/158
5996027	November 1999	Volk et al.	710/13
6006291	December 1999	Rasmussen et al.	710/38
6205516	March 2001	Usami	711/105
6266715	July 2001	Loyer et al.	710/22
6304925	October 2001	Liu et al.	710/62
6378049	April 2002	Stracovsky et al.	711/147

ART-UNIT: 2182

PRIMARY-EXAMINER: Gaffin; Jeffrey

ASSISTANT-EXAMINER: Nguyen; Tanh Q.

ATTY-AGENT-FIRM: Beyer Weaver & Thomas LLP

ABSTRACT:

h e b b g e e e f e e ef b e

A universal access controller is described. The universal resource access controller is coupled to a requesting system and a resource, such that when the requesting system desires access to the resource, the requesting system generates a resource access request which is passed to the universal resource controller. The universal resource controller, in turn, uses a specific characteristic operating parameter of the requested resource as well as a current state of the requested resource to generate a corresponding sequenced universal access request command suitable for accessing the resource as required by the requesting system.

18 Claims, 35 Drawing figures

☐ 4. Document ID: US 6480097 B1

L18: Entry 4 of 22 File: USPT Nov 12, 2002

US-PAT-NO: 6480097
DOCUMENT-IDENTIFIER: US 6480097 B1

TITLE: Security control for personal computer

DATE-ISSUED: November 12, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Zinsky; Timothy R.	Houston	TX		
Shaver; Charles N.	Cypress	TX		
Kaiser, Jr.; Roger A.	Spring	TX		
Rawlins; Paul B.	Spring	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Compaq Information Technologies Group, L.P.	Houston	TX			02

APPL-NO: 09/ 234392 [PALM]
DATE FILED: January 20, 1999

PARENT-CASE:
CROSS-REFERENCE TO RELATED APPLICATIONS This application is a continuation of U.S. application Ser. No. 08/779,061, filed Jan. 6, 1997, now U.S. Pat. No. 5,963,142, which is a file wrapper continuation of U.S. application Ser. No. 08/398,343, filed Mar. 3, 1995, now abandoned, which are incorporated herein for reference.

INT-CL: [07] G06 F 12/14, G11 C 16/22

US-CL-ISSUED: 340/5.8; 713/202, 711/164, 340/5.74
US-CL-CURRENT: 340/5.8; 340/5.74, 711/164, 713/202

FIELD-OF-SEARCH: 340/5.8, 340/5.74, 340/825.5, 713/202, 713/166, 713/194, 713/193, 713/183, 713/2, 705/55, 380/52, 439/628, 439/639, 439/952, 711/147, 711/164

h e b b g e e e f e e ef b e

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
3890601	June 1975	Pietrolewicz	340/172.5
4891838	January 1990	Faber	340/825.34
4942606	July 1990	Kaiser et al.	380/4
4959860	September 1990	Watters et al.	380/4
5060263	October 1991	Bosen et al.	340/825.31
5173940	December 1992	Lantz et al.	380/25
5212729	May 1993	Schafer	705/55
5265163	November 1993	Golding et al.	713/202
5313639	May 1994	Chao	380/25
5355414	October 1994	Hale et al.	380/25
5375243	December 1994	Parzych et al.	395/725
5377343	December 1994	Yaezawa	380/3
5388156	February 1995	Balckledge, Jr. et al.	713/202
5451934	September 1995	Dawson et al.	340/825.31
5475762	December 1995	Morisawa et al.	340/825.31
5533125	July 1996	Bensimon et al.	380/4
5537544	July 1996	Morisawa et al.	380/25

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
2 154 344	September 1985	GB	380/25

OTHER PUBLICATIONS

Compaq DeskPro/M Reference Guide, Compaq Computer Corporation, pp. 5-1 through 5-13.
Fastlock User's Manual, Version 1.0, Rupp Corp., New York, New York, pp. 1-8.
Disklock Advertisement, PC Magazine, vol. 10, No. 11, Jun. 11, 1991, p. 139.

ART-UNIT: 2635

PRIMARY-EXAMINER: Horabik; Michael

ASSISTANT-EXAMINER: Bangachon; William

ATTY-AGENT-FIRM: Akin Gump Stauss Hauer & Feld LLP

ABSTRACT:

A personal computer provides security features enabling control over access to data retained in the computer. The computer is secured by having the system ROM provide a password at power-on to a security device which controls access to the secured features. Once a password has been downloaded to the security device, a Protect Resources command is issued to the security device. To gain access to the secured

h e b b g e e e f e e ef b e

feature after boot-up, the user provides the correct password to the security device and waits for approval from the security device. Since the security device only verifies the password and does not divulge it, security of the system is enhanced. Once access to protected resources is no longer required, the computer issues another Protect Resources command to the security device to once more lock access to the protected resources.

18 Claims, 15 Drawing figures

☐ 5. Document ID: US 6393455 B1

L18: Entry 5 of 22 File: USPT May 21, 2002

US-PAT-NO: 6393455
DOCUMENT-IDENTIFIER: US 6393455 B1
**** See image for Certificate of Correction ****

TITLE: Workload management method to enhance shared resource access in a multisystem environment

DATE-ISSUED: May 21, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Eilert; Catherine K.	Wappingers Falls	NY		
Yocom; Peter B.	Wappingers Falls	NY		
King; Gary M.	Millbrook	NY		
Aman; Jeffrey D.	Poughkeepsie	NY		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corp.	Armonk	NY			02

APPL-NO: 08/ 827528 [PALM]
DATE FILED: March 28, 1997

INT-CL: [07] G06 F 9/00

US-CL-ISSUED: 709/105; 709/104
US-CL-CURRENT: 718/105; 718/104

FIELD-OF-SEARCH: 709/100, 709/101, 709/102, 709/105, 709/104, 709/108, 709/103, 709/201, 709/226, 709/228, 395/709, 395/710, 714/48

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL

h e b b g e e e f e e ef b e

3702462	November 1972	England	340/172.5
4177513	December 1979	Hoffman et al.	364/200
4858108	August 1989	Ogawa et al.	364/200
5008808	April 1991	Fries et al.	364/200
5031089	July 1991	Liu et al.	364/200
5220653	June 1993	Miro	395/275
5301323	April 1994	Maeurer et al.	395/650
5379381	January 1995	Lamb	395/275
5416921	May 1995	Frey et al.	395/575
5421011	May 1995	Camillone et al.	709/100
5446737	August 1995	Cidon et al.	370/85.5
5452455	September 1995	Brown et al.	395/700
5459864	October 1995	Brent et al.	395/650
5473773	December 1995	Aman et al.	709/104
5504894	April 1996	Ferguson et al.	395/650
5507032	April 1996	Kimura	395/826
5537542	July 1996	Eilert et al.	395/184.01
5603029	February 1997	Aman et al.	395/675
5675739	October 1997	Eilert et al.	709/226
5819047	October 1998	Bauer et al.	395/200.59

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
07-244629	September 1995	JP	

ART-UNIT: 2151

PRIMARY-EXAMINER: Banankhah; Majid

ATTY-AGENT-FIRM: Kinnaman, Jr., Esq.; William A. Heslin & Rothenberg, P.C. Radigan, Esq.; Kevin P.

ABSTRACT:

A technique is disclosed for managing a workload distributed across multiple data processing systems to enhance shared resource access to meet a common performance standard. The technique includes on at least one system, measuring performance of the work units on the system to create local performance data, and on at least some of the systems sending the local performance data to at least one other system of the multiple data processing systems. The method further includes on at least one of the systems, receiving the performance data from the sending systems to create remote performance data, and adjusting at least one control parameter for accessing shared resources in response to the local and remote performance data to modify the performance of the work units distributed across the data processing systems to achieve the common performance standard. A dynamic resource clustering process is also employed to enhance the shared resource management.

62 Claims, 15 Drawing figures

☐ 6. Document ID: US 6374399 B1

L18: Entry 6 of 22 File: USPT Apr 16, 2002

US-PAT-NO: 6374399
DOCUMENT-IDENTIFIER: US 6374399 B1

TITLE: Apparatus and method for providing list and read list capability for a host computer system

DATE-ISSUED: April 16, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Mann; Daniel	Austin	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Advanced Micro Devices, Inc.	Sunnyvale	CA			02

APPL-NO: 09/ 295977 [PALM]
DATE FILED: April 21, 1999

INT-CL: [07] G06 F 9/44

US-CL-ISSUED: 717/4; 717/4, 717/5, 717/7
US-CL-CURRENT: 717/127; 717/136

FIELD-OF-SEARCH: 717/7, 717/4, 717/9, 717/5, 715/5, 714/38

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5493675	February 1996	Faiman, Jr. et al.	395/700
5815653	September 1998	You et al.	717/4
5819093	October 1998	Davidson et al.	395/704
5892941	April 1999	Khan et al.	717/4

OTHER PUBLICATIONS

Programmer Reference, "TLA Programmatic Interface (TPI)/TLA 700 Series Logic Analyzer", Tektronix, Inc., WIlsonville, OR, pp. 1-120 (admitted prior to Apr. 21, 1999).

ART-UNIT: 2122

PRIMARY-EXAMINER: Dam; Tuan Q.

h e b b g e e e f e e ef b e

ASSISTANT-EXAMINER: Kendall; Chuck O

ATTY-AGENT-FIRM: Zagorin, O'Brien & Graham, LLP

ABSTRACT:

A host computer system is coupled to a target computer system in a computer system debug environment and accesses selected resources. A first function executed on the host, encodes a data structure with target resource descriptor information for at least one of the selected resources. A second function accesses the selected resources in accordance with the target resource descriptor information encoded in the data structure.

23 Claims, 4 Drawing figures

☐ 7. Document ID: US 6246987 B1

L18: Entry 7 of 22 File: USPT Jun 12, 2001

US-PAT-NO: 6246987
DOCUMENT-IDENTIFIER: US 6246987 B1

TITLE: System for permitting access to a common resource in response to speaker identification and verification

DATE-ISSUED: June 12, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Fisher; Thomas D.	Plano	TX		
Mowry; Dearborn R.	Irving	TX		
Spiess; Jeffrey J.	Lewisville	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Alcatel USA Sourcing, L.P.					02

APPL-NO: 09/ 018433 [PALM]
DATE FILED: February 4, 1998

INT-CL: [07] G10 L 17/00

US-CL-ISSUED: 704/273; 704/247, 704/252
US-CL-CURRENT: 704/273; 704/247, 704/252

FIELD-OF-SEARCH: 704/273, 704/275, 704/270, 704/246, 704/247, 704/248, 704/252

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

h e b b g e e e f e e ef b e

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
3752904	August 1973	Waterbury	348/13
4363102	December 1982	Holmgren et al.	704/238
5091947	February 1992	Ariyoshi et al.	704/246
5165095	November 1992	Borcherding	379/88.03
5212832	May 1993	Ness-Cohn	455/514
5297183	March 1994	Bareis et al.	455/410
5375244	December 1994	McNair	710/200
5430827	July 1995	Rissanen	704/272
5719921	February 1998	Vysotsky et al.	379/88.01
5758317	May 1998	Peterson et al.	704/247
5758322	May 1998	Rongley	704/275
5832063	November 1998	Vysotsky et al.	379/88.03

ART-UNIT: 261

PRIMARY-EXAMINER: Korzuch; William R.

ASSISTANT-EXAMINER: Azad; Abul K.

ATTY-AGENT-FIRM: Anderson, Levine & Lintel,

ABSTRACT:

A method of accessing a resource, where the resource is accessible by a plurality of authorized persons. The method receives a signal representative of a first utterance from a first person, wherein the first utterance represents information. In response to the signal representative of a first utterance from a first person, the method determines with a computer system whether the first person is one of the plurality of authorized persons. The method also receives a signal representative of a first utterance from a second person, where the signal representative of a first utterance from a second person represents the same information as the information represented by the signal representative of a first utterance from the first person. In response to the signal representative of the first utterance from the second person, the method determines with the computer system whether the second person is one of the plurality of authorized persons.

32 Claims, 4 Drawing figures

☐ 8. Document ID: US 6195676 B1

L18: Entry 8 of 22 File: USPT Feb 27, 2001

US-PAT-NO: 6195676
DOCUMENT-IDENTIFIER: US 6195676 B1

TITLE: Method and apparatus for user side scheduling in a multiprocessor operating system program that implements distributive scheduling of processes

h e b b g e e e f e e ef b e

DATE-ISSUED: February 27, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Spix; George A.	Eau Claire	WI		
Wengelski; Diane M.	Eau Claire	WI		
Hawkinson; Stuart W.	Eau Claire	WI		
Johnson; Mark D.	Eau Claire	WI		
Burke; Jeremiah D.	Eau Claire	WI		
Thompson; Keith J.	Eau Claire	WI		
Gaertner; Gregory G.	Eau Claire	WI		
Brussino; Giacomo G.	Eau Claire	WI		
Hessel; Richard E.	Altoona	WI		
Barkai; David M.	Eau Claire	WI		
Chen; Steve S.	Chippewa Falls	WI		
Oslon; Steven G.	Chippewa Falls	WI		
Strout, II; Robert E.	Livermore	CA		
Masamitsu; Jon A.	Livermore	CA		
Cox; David M.	Livermore	CA		
O'Gara; Linda J.	Livermore	CA		
O'Hair; Kelly T.	Livermore	CA		
Seberger; David A.	Livermore	CA		
Rasbold; James C.	Livermore	CA		
Cramer; Timothy J.	Pleasanton	CA		
Van Dyke; Don A.	Pleasanton	CA		
Chandramouli; Ashok	Fremont	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Silicon Graphics, Inc.	Mountain View	CA			02

APPL-NO: 08/ 003000 [PALM]
DATE FILED: January 11, 1993

PARENT-CASE:
This is a divisional of application Ser. No. 07/537,466, filed Jun. 11, 1990, now issued as U.S. Pat. No. 5,179,702 filed Dec. 29, 1989 now U.S. Pat. No. 5,197,130, which is a CIP of U.S. Ser. No. 07/459,083.

INT-CL: [07] G06 F 9/46

US-CL-ISSUED: 709/107
US-CL-CURRENT: 718/107

FIELD-OF-SEARCH: 395/375, 395/650, 395/800

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
3593300	July 1971	Driscoll, Jr. et al.	
3614715	October 1971	Podvin	395/650
3648253	March 1972	Mullery et al.	709/100
4073005	February 1978	Parkin	709/104
4099235	July 1978	Hoschler et al.	
4183083	January 1980	Chatfield	
4394727	July 1983	Hoffman et al.	395/650
4484274	November 1984	Berenbaum et al.	
4494188	January 1985	Nakane et al.	
4633387	December 1986	Hartung et al.	
4747130	May 1988	Ho	379/269
4800521	January 1989	Carter et al.	
4809170	February 1989	Leblang et al.	
4837676	June 1989	Rosman	
4845665	July 1989	Heath et al.	
4890257	December 1989	Anthias et al.	
4939507	July 1990	Beard et al.	340/706
4951192	August 1990	Chase, Jr. et al.	
5050070	September 1991	Chastain et al.	712/203

OTHER PUBLICATIONS

Cheriton, David Ross, "Multi-Process Structuring and The Thoth Operating System," Doctorial Thesis, University of Waterloo, 1978, pp. iv, 1-4, 42-51, and 59-62. George S. Almasi, et al.; "Highly Parallel Computing"; Alan Apt, Editor; The Benjamin/Cummings Publishing Company, Inc.; Redwood City, California; 248-276 (1989).

ART-UNIT: 273

PRIMARY-EXAMINER: Ellis; Richard L.

ATTY-AGENT-FIRM: Schwegman, Lundberg, Woessner & Kluth, P.A.

ABSTRACT:

An integrated software architecture for a highly parallel multiprocessor system having multiple tightly-coupled processors that share a common memory efficiently controls the interface with and execution of programs on such a multiprocessor system. The software architecture combines a symmetrically integrated multithreaded operating system and an integrated parallel user environment. The operating system distributively implements an anarchy-based scheduling model for the scheduling of processes and resources by allowing each processor to access a single image of the operating system stored in the common memory that operates on a common set of operating system shared resources. The user environment provides a common visual representation for a plurality of program development tools that provide compilation, execution and debugging capabilities for multithreaded user programs and assumes parallelism as the standard mode of operation.

6 Claims, 59 Drawing figures

☐ 9. Document ID: US 5963142 A

L18: Entry 9 of 22 File: USPT Oct 5, 1999

US-PAT-NO: 5963142
DOCUMENT-IDENTIFIER: US 5963142 A

TITLE: Security control for personal computer

DATE-ISSUED: October 5, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Zinsky; Timothy R.	Houston	TX		
Shaver; Charles N.	Cypress	TX		
Kaiser, Jr.; Roger A.	Spring	TX		
Rawlins; Paul B.	Spring	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Compaq Computer Corporation	Houston	TX			02

APPL-NO: 08/ 779061 [PALM]
DATE FILED: January 6, 1997

PARENT-CASE:
This is a continuation of co-pending application Ser. No. 08/779,061 filed on Jan. 6, 1997 and abandoned Ser. No. 08/398,343, filed on Mar. 3, 1995.

INT-CL: [06] G06 F 12/14

US-CL-ISSUED: 340/825.34; 340/825.31, 380/4, 380/25, 395/188.01, 711/164, 707/9
US-CL-CURRENT: 340/5.74; 707/9, 711/164, 713/183, 713/202

FIELD-OF-SEARCH: 340/825.31, 340/825.34, 380/4, 380/23, 380/25, 380/3, 395/186, 395/188.01, 711/164, 711/163, 707/9, 364/709.05

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
3890601	June 1975	Pietrolewicz	340/172.5
4891838	January 1990	Faber	340/825.34 X
4942606	July 1990	Kaiser et al.	380/4
4959860	September 1990	Watters et al.	380/4
5060263	October 1991	Bosen et al.	340/825.31 X
5173940	December 1992	Lantz et al.	380/25

h e b b g e e e f e e ef b e

5212729	May 1993	Schafer	380/4
5265163	November 1993	Golding et al.	380/25
5313639	May 1994	Chao	380/25 X
5355414	October 1994	Hale et al.	380/25
5375243	December 1994	Parzych et al.	395/725
5377343	December 1994	Yaezawa	380/3 X
5388156	February 1995	Blackledge, Jr. et al.	380/4
5451934	September 1995	Dawson et al.	340/825.31
5475762	December 1995	Morisawa et al.	340/825.31 X
5533125	July 1996	Bensimon et al.	380/4
5537544	July 1996	Morisawa et al.	380/25 X

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
2 154 344	September 1985	GB	380/25

OTHER PUBLICATIONS

COMPAQ DESKPRO/M Reference Guide, pp. 5-1 through 5-13. No Date.
DiskLock Advertisement, PC Magazine, vol. 10, No. 11, Jun. 11, 1991; New York, New York; p. 139.
FastLock User's Manual, Version 1.0, Rupp Corp., New York, New York. No Date.

ART-UNIT: 275

PRIMARY-EXAMINER: Zimmerman; Brian

ASSISTANT-EXAMINER: Wilson, Jr.; William H.

ATTY-AGENT-FIRM: Akin, Gump, Strauss, Hauer & Feld, L.L.P.

ABSTRACT:

A personal computer provides security features enabling control over access to data retained in the computer. The computer is secured by having the system ROM provide a password at power-on to a security device which controls access to the secured features. Once a password has been downloaded to the security device, a Protect Resources command is issued to the security device. To gain access to the secured feature after boot-up, the user provides the correct password to the security device and waits for approval from the security device. Since the security device only verifies the password and does not divulge it, security of the system is enhanced. Once access to protected resources is no longer required, the computer issues another Protect Resources command to the security device to once more lock access to the protected resources.

12 Claims, 15 Drawing figures

☐ 10. Document ID: US 5948089 A

h e b b g e e e f e e ef b e

L18: Entry 10 of 22 File: USPT Sep 7, 1999

US-PAT-NO: 5948089
DOCUMENT-IDENTIFIER: US 5948089 A

TITLE: Fully-pipelined fixed-latency communications system with a real time dynamic bandwidth allocation

DATE-ISSUED: September 7, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wingard; Drew Eric	San Carlos	CA		
Rosseel; Geert Paul	Menlo Park	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Sonics, Inc.	Los Altos	CA			02

APPL-NO: 08/ 924368 [PALM]
DATE FILED: September 5, 1997

INT-CL: [06] G06 F 13/00

US-CL-ISSUED: 710/107; 709/251, 710/111, 710/104
US-CL-CURRENT: 710/107; 709/251, 710/104, 710/111

FIELD-OF-SEARCH: 395/200.81, 395/287, 395/291, 395/284, 395/200.5, 395/200.52, 709/220, 709/222, 709/251, 710/104, 710/111, 710/107, 710/129

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4979096	December 1990	Ueda et al.	395/200.78
5414813	May 1995	Shiobara	395/200.75
5465330	November 1995	Komatsu et al.	395/824
5553245	September 1996	Su et al.	395/284
5701420	December 1997	Nedwek et al.	395/284
5727169	March 1998	Calzi	395/284

ART-UNIT: 271

PRIMARY-EXAMINER: Auve; Glenn A.

ATTY-AGENT-FIRM: Blakely, Sokoloff, Taylor & Zafman LLP

ABSTRACT:

The present invention provides for an on-chip communications method with fully

h e b b g e e e f e e ef b e

distributed control combining a fully-pipelined, fixed-latency, synchronous bus with a two-level arbitration scheme where the first level of arbitration is a framed, time-division-multiplexing arbitration scheme and the second level is a fairly-allocated round-robin scheme implemented using a token-passing mechanism. Both the latency and the bandwidth allocation are software programmable in real-time operation of the system. The present invention also provides for a communications system where access to a shared resource is controlled by the above communications protocol. Access to and from the shared resource from the subsystem is through a bus interface module. The bus interface modules provide a level of indirection between the subsystem to be connected and the shared resource. This allows the decoupling of system performance requirements from subsystem requirements. Communication over the bus is fully memory mapped.

38 Claims, 15 Drawing figures

☐ 11. Document ID: US 5941967 A

L18: Entry 11 of 22 File: USPT Aug 24, 1999

US-PAT-NO: 5941967
DOCUMENT-IDENTIFIER: US 5941967 A

TITLE: Unit for arbitration of access to a bus of a multiprocessor system with multiprocessor system for access to a plurality of shared resources, with temporary masking of pseudo random duration of access requests for the execution of access retry

DATE-ISSUED: August 24, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Zulian; Ferruccio	Milan			IT

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Bull HN Information Systems Italia S.p.A.				IT	03

APPL-NO: 08/ 956218 [PALM]
DATE FILED: October 22, 1997

FOREIGN-APPL-PRIORITY-DATA:

COUNTRY	APPL-NO	APPL-DATE
EP	96830621	December 13, 1996

INT-CL: [06] G06 F 13/36

US-CL-ISSUED: 710/107; 710/118, 710/240
US-CL-CURRENT: 710/107; 710/118, 710/240

FIELD-OF-SEARCH: 395/287, 395/285, 395/293, 395/728, 395/729, 395/298, 710/107, 710/105, 710/113, 710/240, 710/241, 710/118

h e b b g e e e f e e ef b e

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5706446	January 1998	Kalish et al.	395/293
5774679	June 1998	Kondo et al.	395/285
5781745	July 1998	Ramelson et al.	395/293

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
665 501	August 1995	EP	
96 35175	November 1996	WO	

OTHER PUBLICATIONS

European Search Report for Application No. EP 96 83 0621.

ART-UNIT: 271

PRIMARY-EXAMINER: Ray; Gopal C.

ATTY-AGENT-FIRM: Manzo; Edward D. Murphy; Mark J.

ABSTRACT:

In a multiprocessor system with shared resources, which several processors access via a system bus by presenting bus access requests to an arbitration unit and receiving from the latter access grant signals and in which the busy state of a resource or a conflict of consistency determine the generation of a RETRY signal and compel the processor, which has obtained access to the bus, to execute an access RETRY attempt, consecutive repeated access RETRY attempts of the same processor activate logic of the arbitration unit which temporarily mask, for a varying duration, the access requests of the same processor for the execution of further consecutive RETRY attempts, the varying duration first increasing as a function of the number of further RETRY attempts and then varying in a random manner.

7 Claims, 11 Drawing figures

☐ 12. Document ID: US 5893157 A

L18: Entry 12 of 22 File: USPT Apr 6, 1999

US-PAT-NO: 5893157
DOCUMENT-IDENTIFIER: US 5893157 A

TITLE: Blocking symbol control in a computer system to serialize accessing a data

h e b b g e e e f e e ef b e

resource by simultaneous processor requests

DATE-ISSUED: April 6, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Greenspan; Steven Jay	Hyde Park	NY		
Scalzi; Casper Anthony	Poughkeepsie	NY		
Plambeck; Kenneth Ernest	Poughkeepsie	NY		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 864402 [PALM]
DATE FILED: May 28, 1997

INT-CL: [06] G06 F 12/00

US-CL-ISSUED: 711/150; 711/151, 711/152
US-CL-CURRENT: 711/150; 711/151, 711/152

FIELD-OF-SEARCH: 711/150, 711/151, 711/152

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4851990	July 1989	Johnson et al.	395/280
5081572	January 1992	Arnold	711/163
5142676	August 1992	Fried et al.	711/152
5333297	July 1994	Lemaire et al.	395/500
5410697	April 1995	Baird et al.	711/152
5488729	January 1996	Vegesna et al.	395/385
5574922	November 1996	James	395/561
5590326	December 1996	Manabe	711/150
5623671	April 1997	Ando et al.	395/726
5636361	June 1997	Ingerman	711/150
5659711	August 1997	Sugita	711/144
5669002	September 1997	Buch	395/726
5696939	December 1997	Iacobovici et al.	711/150

ART-UNIT: 278

PRIMARY-EXAMINER: Lim; Krisna

ATTY-AGENT-FIRM: Ehrlich; Marc A. Goldman; Bernard M.

h e b b g e e e f e e ef b e

ABSTRACT:

PLO (perform locked operation) instructions containing blocking symbols are executed on each of multiple processors in a computer system to control coherence in data structures which may be changed by any of multiple processors in a computer system. The blocking symbol is extracted from a PLO instruction instance when invoked by its executing processor. Then the processor hashes the blocking symbol using hardware-microcode (H-M) to generate the location of a lock field in protected storage. The PLO instruction's blocking symbol is associated with a computer resource unit by software providing the PLO instruction, and the blocking symbol then associates the resource with a protected lock through the hashing operation on the blocking symbol. A processor must obtain the lock for a blocking symbol before the executing PLO instruction instance is allowed to make access and change the resource unit associated with the blocking symbol. The blocking symbol controls the PLO operations by serializing simultaneously PLO instruction access requests being made by multiple processors to the same resource unit using the same blocking symbol to allow only one PLO instruction instance to have exclusive access to the resource at a time.

18 Claims, 7 Drawing figures

☐ 13. Document ID: US 5887167 A

L18: Entry 13 of 22 File: USPT Mar 23, 1999

US-PAT-NO: 5887167
DOCUMENT-IDENTIFIER: US 5887167 A

TITLE: Synchronization mechanism for providing multiple readers and writers access to performance information of an extensible computer system

DATE-ISSUED: March 23, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Sutton; Carl D.	Palo Alto	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Apple Computer, Inc.	Cupertino	CA			02

APPL-NO: 08/ 553104 [PALM]
DATE FILED: November 3, 1995

INT-CL: [06] G06 F 9/46

US-CL-ISSUED: 395/676; 395/677, 395/680
US-CL-CURRENT: 719/314; 718/106, 718/108

FIELD-OF-SEARCH: 395/677, 395/680, 395/676

PRIOR-ART-DISCLOSED:

h e b b g e e e f e e ef b e

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
3818458	June 1974	Deese	340/172.5
3906454	September 1975	Martin	340/172.5
4849879	July 1989	Chinnaswamy et al.	364/200
5220562	June 1993	Takada et al.	370/85.13
5301290	April 1994	Tetzlaff et al.	395/425
5305448	April 1994	Insalaco et al.	395/425
5347649	September 1994	Alderson	395/600
5379406	January 1995	Wade	395/500
5440545	August 1995	Buchholz et al.	370/60
5485574	January 1996	Bolosky et al.	395/183.11
5555396	September 1996	Alferness et al.	395/474
5581482	December 1996	Wiedenman et al.	364/550

OTHER PUBLICATIONS

Kleiman et al, Symmetric Multiprocessing In Solaris 2.0, Compcon Spring 92 San Francisco CA p. 181 1992.
F. Fotouhi et al., "The Generalized Index Model for Object-Oriented Database Systems", IEEE, 1991 (pp. 302-308).
E. Horowitz et al., "Fundamentals of Computer Algorithms", 1978 (pp. 48,58,59 & 63).

ART-UNIT: 275

PRIMARY-EXAMINER: Toplu; Lucien U.

ATTY-AGENT-FIRM: Cesari and McKenna, LLP

ABSTRACT:

A synchronization arrangement provides writer and reader entities access to an information resource, such as a trace buffer, located in a registry of a computer. The arrangement comprises a counter upon which atomic increments are performed to allocate entries of the trace buffer for temporarily storing trace message fragments provided by the writer entities. The arrangement also comprises reassembly queues for temporarily storing the message fragments sequentially retrieved from the trace buffer by the reader entities.

17 Claims, 9 Drawing figures

☐ 14. Document ID: US 5586289 A

L18: Entry 14 of 22 File: USPT Dec 17, 1996

US-PAT-NO: 5586289
DOCUMENT-IDENTIFIER: US 5586289 A

h e b b g e e e f e e ef b e

TITLE: Method and apparatus for accessing local storage within a parallel processing computer

DATE-ISSUED: December 17, 1996

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chin; Danny	Princeton Jct.	NJ		
Peters, Jr.; Joseph E.	East Brunswick	NJ		
Taylor; Herbert H.	Pennington	NJ		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
David Sarnoff Research Center, Inc.	Princeton	NJ			02

APPL-NO: 08/ 228465 [PALM]
DATE FILED: April 15, 1994

INT-CL: [06] G06 F 13/14

US-CL-ISSUED: 395/438; 395/478, 395/806, 364/230, 364/238.4, 364/242.3, 364/242.91, 364/DIG.1
US-CL-CURRENT: 711/111; 711/151, 712/16

FIELD-OF-SEARCH: 395/800, 395/200.01, 395/200.13, 395/200.15, 395/841, 395/298, 395/305, 395/478, 395/494, 395/559, 395/860, 395/871, 395/298, 395/305, 395/427, 395/438, 365/189, 364/131-134

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4344134	August 1982	Barnes	364/200
4412286	October 1983	O'Dowd	364/200
4414624	November 1983	Summer	364/200
4636944	January 1987	Hodge	364/200
4716525	December 1987	Gilanyi et al.	364/200
4736363	April 1988	Aubin et al.	370/60
4740954	April 1988	Cotton et al.	370/60
4796232	January 1989	House	365/189
4809362	February 1989	Claus et al.	455/607
4837676	June 1989	Rosman	364/200
4866668	September 1989	Edmonds et al.	364/900
4965718	October 1990	George	364/200
5018133	May 1991	Tsukakoshi et al.	370/16
5038282	August 1991	Gilbert et al.	395/200.01
5105424	April 1992	Flaig et al.	370/94.1
5109379	April 1992	Kume et al.	370/94.3
5113523	May 1992	Colley	395/800

h e b b g e e e f e e ef b e

5117430	May 1992	Berglund	370/85.1
5121502	June 1992	Rau	395/8
5125076	June 1992	Faber et al.	395/200
5129077	July 1992	Hillis	395/800
5175865	December 1992	Hillis	395/800
5197140	March 1993	Balmer	395/400
5388217	February 1995	Benzschawel et al.	395/856

OTHER PUBLICATIONS

S. B. Wu and M. T. Liu, "A Cluster Structure as an Interconnection Network for Large Multimicrocomputer Systems" IEEE Transactions on Computers, vol. 3-30, pp. 254-264 (Apr. 1981).
F. Pittelli and D. Smitley, "Analysis of a 3D Toroidal Network for a Shared Memory Architecture" Proceedings of Supercomputing '88 (IEEE Computer Society Press, Los Alamitos, CA; 1988) pp. 42-47.
C. Germain et al. "An Interconnection Network and a Routing Scheme for a Massively Parallel Multicomputer" Third Symposium on the Frontiers of Massiv ely Parallel Computation, Proceedings, 9 IEEE Computer Society Press; Los Alamitos CA; 1990 pp. 368-371.
"Maxion" Multiprocessor System product literature, (Concurrent Computer Corporation; Oceanport, New Jersey; Oct. 1993) 33 pages.
C. A. Robinson, Jr., "Advanced High-Speed Chips Move to Parallel Processing" Signal, Jun. 1993, pp. 23-31.
Z. G. Vranesic et al., "Hector: A Hierarchically Structured Shared-Memory Microprocessor" Computer, Jan. 1991, pp. 72-79.
R. Alverson et al, Conference Proceedings, 1990 Conference On Supercomputing, "The Tera Computer System" Association for Computing Machinery, 1990) pp. 1-6.

ART-UNIT: 232

PRIMARY-EXAMINER: Shah; Alpesh M.

ATTY-AGENT-FIRM: Burke; William J.

h e b b g e e e f e e ef b e

L18: Entry 13 of 22 File: USPT Mar 23, 1999

DOCUMENT-IDENTIFIER: US 5887167 A
TITLE: Synchronization mechanism for providing multiple readers and writers access to performance information of an extensible computer system

Abstract Text (1):
A synchronization arrangement provides writer and reader entities access to an information resource, such as a trace buffer, located in a registry of a computer. The arrangement comprises a counter upon which atomic increments are performed to allocate entries of the trace buffer for temporarily storing trace message fragments provided by the writer entities. The arrangement also comprises reassembly queues for temporarily storing the message fragments sequentially retrieved from the trace buffer by the reader entities.

Brief Summary Text (20):
Briefly, the invention relates to a synchronization arrangement for providing writer and reader entities access to an information resource, such as a trace buffer, located in a registry of a computer. The novel arrangement generally comprises a counter upon which atomic increments are performed to allocate entries of the trace buffer for temporarily storing trace message fragments provided by the writer entities. The arrangement further comprises reassembly queues for temporarily storing the message fragments sequentially retrieved from the trace buffer by the reader entities.

Brief Summary Text (24):
The reader entities are generally responsible for detecting lost, incomplete or missed messages with respect to the trace buffer; preferably, the reassembly queue, in addition to a time-out mechanism, are used to detect these conditions. According to an aspect of the invention, writer entities always write fragments into the buffer in order so that missing fragments (other than the last fragment) can be easily detected. A missing last fragment may be detected using the time-out mechanism comprising a predetermined threshold established with respect to the contents of the counter. If the contents of the counter exceed this threshold before all the fragments of a message are retrieved by a reader, then the reader may assume that the fragment (and, thus, the message) is irretrievably lost.

Detailed Description Text (12):
FIG. 4 is a block diagram of the registry 400 embodying a hierarchical tree data structure architecture of nodes for managing and organizing performance information, such as statistics and tracing, collected by writer entities 320. Each node represents a single data item with a single data type; in addition, data collection presented by each node of the tree can be independently enabled and disabled. The multi-linked tree structure is located within main memory 114 (FIG. 1) having a plurality of address locations for accessing the performance information. An example of a registry is provided in copending and commonly assigned U.S. Patent Application titled Extensible Performance Statistics and Tracing Registration Architecture, filed on even date herewith, which application is incorporated by reference as though fully set forth herein.

Detailed Description Text (17):
The writer entities 320 "register" their intent to collect and store performance

information in registry 400 by creating objects as nodes organized within the tree structure. Specifically, the writer entities 320 are designated as various portions of the tree structure and the information they collect are also described as object nodes of the tree. Each object node is preferably named according to the performance information it collects and, also, the type of data with which it is associated. For example, the root object node 402 of registry 400 is named Root and has an associated counter (Counter) data type.

Detailed Description Text (29):
Coordination of updates of the collected statistics may be effected using atomic instructions for particular information collection data types. For example when updating a statistic using a counter data type, an atomic operation, e.g., a read-modify-write operation, is required because the counter has a current value. If the update involves incrementing that value, the contents of the counter must be read prior to adding one thereto, and the result is then written back to the counter. Since there may be multiple writer entities, the atomic operation basically "locks" the counter to avoid contention. The atomic operation preferably uses a hardware processor as the locking mechanism for the data item.

Detailed Description Text (38):
In accordance with the invention, the trace synchronization arrangement is provided that allows writer and reader entities access to an information resource, such as a trace buffer, located in the registry. FIG. 7 is a schematized block diagram of the trace synchronization arrangement 700 comprising a fragment allocation counter 702 upon which atomic increments are performed by writer entities to allocate entries of a trace buffer 710 for temporarily storing trace message fragments provided by the writer entities.

Detailed Description Text (39):
The counter 702 is preferably a continuous rolling "up" counter whose contents comprises, e.g, a 32-bit word. A writer entity is allocated an entry in the buffer in response to performing an atomic increment on the counter. An atomic increment is, e.g., a read-modify-write atomic operation that effectively "locks" the counter from access by other entities as the writer increments the contents of the counter. Specifically, the writer entity that gains control of the counter increments its contents and performs a modulo operation on the resulting contents; this indicates which entry of the trace buffer has been allocated. The writer then copies a fragment of the message into that buffer entry.

Detailed Description Text (41):
Writer entities preferably allocate message sequence numbers using atomic increments of a global sequence counter 732, which is different from the trace buffer fragment allocation counter 702. In addition, a sequence number of "0" is invalid and never allocated to a writer entity.

Detailed Description Text (42):
Operationally, the writer entity increments the fragment allocation counter 702, thereby allocating one of the entries 712 of the buffer 710. For N entries, a particular entry may described by an atomic increment of the counter, modulo-N (which is the number of entries in the entire trace buffer):

Detailed Description Text (44):
(i) allocate a message sequence number using the sequence counter 732;

Detailed Description Text (45):
(ii) allocate an entry 712 of the trace buffer 710 using the fragment allocation counter 702;

Detailed Description Text (53):
The novel arrangement also provides a means for avoiding allocation of a particular

entry by multiple writers simultaneously accessing the buffer when the fragment allocation counter 702 advances more than N entries. Assume a writer entity is interrupted in the midst of copying fragments into the buffer and the period of interruption is sufficient to allow allocation of N additional entries to other writers. The next N+1 entry is already being copied into by the interrupted writer entity; however, this entry will also be allocated to a different writer entity, causing the two writers to collide.

Detailed Description Text (57):
As noted, the reader entities are generally responsible for detecting lost, incomplete or missed messages with respect to the trace buffer; preferably, the reassembly queue, in addition to a time-out mechanism, are used to detect these conditions. According to an aspect of the invention, since writer entities always write fragments in order, fragments (other than the last fragment) that are missing can be easily detected. A missing last fragment may be detected using the time-out mechanism comprising a predetermined threshold established with respect to the contents of the counter 702. Specifically, if the contents of the counter exceed the threshold before all the fragments of a message are retrieved by a reader, then the reader may assume that the message is irretrievably lost. In the illustrative embodiment described herein, the predetermined threshold is preferably half the number of entries.

Detailed Description Text (58):
For example, assume there are 100 entries (with sequence numbers 1-100) in the trace buffer and the first fragment that is retrieved by a reader entity is fragment number 5 of message sequence number 50. If the counter 702 progresses to message sequence number 100 and the reader has yet to retrieve fragment numbers 1-4 of message sequence number 50, those fragments are presumed lost and that message is incomplete.

Detailed Description Text (59):
Moreover, each reader maintains a private copy of the contents of the fragment allocation counter 702. As a fragment is retrieved from the trace buffer, this private copy is compared with the contents of that counter. If the difference between the resulting values is greater than N (the number of trace buffer entries), the reader has missed at least one entire set of trace buffer fragments. Accordingly, the reader starts over with the oldest fragment in the buffer. The oldest fragment in a trace buffer may be calculated as follows:

Detailed Description Text (68):
Specifically, reader entities poll for new message fragments using their private copies of the fragment allocation counter 702 to detect when they have read as many fragments as possible. If a private copy matches the counter contents, the reader has reached the youngest fragment in the trace buffer and that reader entity transitions to an "idle" state. In addition, if a reader encounters a "0"-valued sequence number in a fragment, the reader knows a writer was interrupted and proceeds to transition to its idle state (subject, of course, to the time-out mechanism described above). Finally, if a sequence number is encountered in the youngest fragment that indicates the fragment is "older" than the time-out period, then a writer entity was interrupted between allocating a message sequence number and "zeroing" the sequence number in the fragment entry; accordingly, the reader can transition to its idle state.

CLAIMS:

1. A synchronization arrangement for use by reader and writer entities executing on a computer having a memory, the synchronization arrangement comprising:

an information resource located in the memory of the computer, the information resource having a plurality of entries and being accessible by the reader and

writer entities;

a first counter operatively coupled to the writer entities for having atomic increments performed thereon, the first counter configured to allocate a specific entry of the information resource to a given writer entity in response to the performance of an atomic increment by the given writer entity, the resource entries temporarily storing messages sequentially posted by the writer entities;

means for limiting allocation of the entries of the information resource to the writer entities except through the first counter; and

a reassembly queue for temporarily storing messages sequentially retrieved from the resource by the reader entities,

wherein only one writer entity, at any given time, is allocated a specific entry of the information resource.

2. The synchronization arrangement of claim 1 further comprising a time-out mechanism for detecting incomplete messages retrieved from the resource, the time-out mechanism comprising a predetermined threshold established by contents of the first counter, wherein a message is incomplete if it is not completely retrieved by a reader entity before the contents of the first counter exceed the predetermined threshold.

4. The synchronization arrangement of claim 3 wherein each fragment contains a fragment number and a sequence number, the sequence number being sequentially assigned to each message prior to being apportioned into fragments by a second counter.

5. The synchronization arrangement of claim 4 wherein the first counter is a fragment allocation counter and the second counter is a sequence allocation counter, and wherein a sequence number of zero is an invalid sequence number that signifies copying of a fragment into a trace buffer entry.

9. A method for providing reader and writer entities access to a trace buffer located in a memory of a computer to reliably exchange complete trace messages, the method comprising the steps of:

allocating entries of the trace buffer to the writer entities using a counter responsive to atomic increments thereon, the trace buffer entries temporarily storing trace messages sequentially provided by the writer entities;

temporarily storing the trace messages sequentially retrieved from the trace buffer by the reader entities using a reassembly queue; and

detecting incomplete trace messages retrieved from the trace buffer using a time-out mechanism comprising a predetermined threshold established by contents of the counter, wherein a trace message is incomplete if it is not completely retrieved by a reader entity before the contents of the counter exceed the predetermined threshold.

15. The synchronization arrangement of claim 1 wherein the first counter defines corresponding contents and a writer entity, having control of the first counter, either increments or decrements its corresponding contents and performs an operation on the resulting contents to identify the entry of the information resource being allocated.